Operační systém CP/M

Ing. Karel Richta, CSc.

Praha 1991
SNTL
Nakladatelství
technické
literatury

Kniha je rozdělena do tří hlavních částí. První část (kap. 2) je věnována popisu struktury a funkce osobních počítačů. Druhá část se zabývá vlastním operačním systémem, jeho uživatelským a programovým prostředím a vazbou na technické prostředky (kap. 3 až 6). V závěru jsou základní informace shrnuty do tabulek uvedených jako příloha (kap. 7).

Kniha je určena všem uživatelům osobních počítačů a zájemcům o tuto oblast výpočetní techniky. Mohou ji využít i posluchači a učitelé vysokých škol i ostatní pracovníci přicházející do styku s osobními počítači.

Lektorovali: RNDr. J. Brodský, CSc., Ing. E. Tschernoster

Redakce teoretické literatury
Hlavní redaktorka: RNDr. Blanka Kutinová, CSc.
Odpovědný redaktor: Jan Beneš

ISBN 80-03-00519-1

Obsah

1 Úvod

Rozvoj technologie polovodičových součástek a jejich miniaturizace dovolily uvažovat na počátku sedmdesátých let o stavbě mikropočítačových systémů. Nízká cena potřebných obvodů umožnila využití mikropočítačových systémů v nejrozmanitějších aplikacích. Široké spektrum mikropočítačových systémů lze rozdělit na dva typické okruhy:

- specializované systémy, určené např. pro řízení v reálném čase, ovládání specializovaných subsystémů (databázové procesory) apod.,
- univerzální systémy.

Specializované systémy tvoří jeden z nejdůležitějších okruhů nasazení mikropočítačové techniky. Z hlediska uživatele však představují poměrně uzavřené celky, zaměřené na jedinou aplikaci. Naproti tomu *univerzální systémy* dovolují využití mikropočítače v řadě aplikací. Tato pružnost předurčuje univerzální systémy pro mnohem větší spektrum uživatelů, což zpětně vyvolává rychlý rozvoj jak technických prostředků, tak programového vybavení. Velká produkce dále snižuje cenu univerzálních systémů, tím je umožněno, aby byl celý univerzální mikropočítačový systém používán jedinou osobou.

Označení *osobní počítač* odráží právě tento rys univerzálních mikropočítačových systémů. Takto obecně pojatý význam označení zahrnuje však příliš velkou třídu systémů, které se značně liší svými výpočetními schopnostmi. Z hlediska výkladu se zdá výhodnější rozdělit tuto třídu do kategorií, charakterizovaných určitými výpočetními schopnostmi. Nejvážnějšími faktory pro klasifikaci budou zřejmě dostupnost dostatečného repertoáru programů a dat, rychlost přístupu k nim a rychlost zpracování. Univerzální mikropočítačové systémy lze podrobněji rozčlenit na následující kategorie:

– kapesní počítače (bez vnější paměti),
– domácí počítače (jen pomalá vnější paměť – obvykle kazetová páska),
– osobní počítače (rychlá vnější paměť – obvykle disk),
– pracovní stanice (špičkové mikropočítače s velkou výpočetní kapacitou, případně s možností využívání prostředků větších systémů).

V dalším textu se zaměříme především na osobní počítače v tomto užším smyslu, neboť tato kategorie představuje nejrozšířenější třídu profesionálně použitelných univerzálních mikropočítačových systémů.

Samotná možnost konstrukce technického zařízení, vybaveného mikroprocesorem a dalšími obvody, neznamená ještě vytvoření skutečně použitelného mikropočítačového systému. Teprve programy vdechují život do chladného prostředí techniky – přesněji řečeno, způsobují dostatečně účelné chování systému. Pro počáteční stadium vývoje je typické základní programové vybavení (nazývané obvykle *monitor* a umístěné v pevné paměti), které dovoluje programování na úrovni strojového kódu.

Programování na úrovni strojového kódu znamená samozřejmě krok zpět oproti stávající úrovni. Již od počátků existovaly proto snahy implementovat na nové třídě počítačů vyšší programovací jazyky. Vhodným kandidátem se zprvu stal programovací jazyk Basic, neboť jeho interpret se podařilo umístit do pevné paměti.

Nový impuls pro další rozvoj programových prostředků přináší doplnění konfigurace mikropočítače o kazetovou magnetickou pásku, dovolující již pro kategorii domácích počítačů vytváření rozmanitých aplikačních programů. Rozumné využívání programových prostředků však obvykle naráželo na potřebu vnějších pamětí s větší kapacitou a rychlejším přístupem. Pro vývoj mikropočítačů bylo velmi důležité, že prakticky ve stejné době, kdy byly k dispozici potřebné integrované obvody, vzniká nový typ levné vnější paměti – pružný disk, který představoval ideální médium pro mikropočítačový systém. Tím vznikly fyzické předpoklady pro realizaci skutečně univerzálního počítačového systému, vybaveného diskovým operačním systémem – typického představitele kategorie osobních počítačů.

Osobní počítače dělíme dále podle šířky dat zpracovaných mikroprocesorem na 8bitové, 16bitové a 32bitové. Typickému základnímu programovému vybavení 16bitových osobních počítačů je věnována publikace [28]. Dalšími praktickými standardy operačních systémů pro osobní počítače a

pracovní stanice jsou systémy OS/2, Unix a Finder. Operačnímu systému Unix je věnována samostatná publikace [3].

Tato kniha se zabývá operačním systémem CP/M a chce dát čtenáři postačující informace i pro zvládnutí systému podobného typu.

V kapitole 2 je stručně probráno technické vybavení 8bitových osobních počítačů. Čtenář v ní nalezne informace o konstrukci osobních počítačů, potřebné z hlediska pochopení prostředí pro práci operačního systému. Čtenáři, který se blíže zajímá o problematiku konstrukce mikropočítačových systémů, lze doporučit například publikaci [27] nebo [24].

Kapitola 3 obsahuje úvod do operačního systému CP/M. Zabývá se historií vzniku systému CP/M, jeho celkovou koncepcí a strukturou.

Uživatelskému prostředí systému CP/M je věnována kapitola 4. Přibližuje čtenáři řídicí jazyk systému CP/M a služební programy dodávané jako součást systému.

Programové prostředí poskytované systémem CP/M je podrobně probráno v kapitole 5. Tato kapitola je určena čtenářům, kteří se zajímají o tvorbu programů pro systém CP/M.

Kapitola 6 se zabývá vazbou systému CP/M na technické prostředí a jsou zde vyloženy základní principy konstrukce technicky závislých složek systému. Součástí kapitoly 6 jsou i informace potřebné pro úpravy, generování a instalaci systému CP/M.

Text je doplněn o přílohy obsahující tabulky potřebné při práci s operačním systémem CP/M.

Kniha je zaměřena speciálně na operační systém CP/M a neobsahuje proto některé partie, potřebné z hlediska obecné teorie operačních systémů. Jedná se zejména o principy konstrukce multiuživatelských a multiprogramových systémů. Čtenář, který se podrobněji zajímá o obecnou problematiku operačních systémů, nalezne příslušné informace např. v publikacích [23], [12] atd.

Od čtenáře se nepředpokládají speciální znalosti z teorie operačních systémů a jen minimální znalosti o technických prostředcích mikropočítačů. Předpokládá se, že čtenář je zběžně seznámen se základními principy programování, ať již ve vyšším programovacím jazyce, neboť některé algoritmy jsou prezentovány ve formě připomínající jazyk Pascal, či v jazyce symbolických instrukcí, který je využit pro detailní prezentaci technicky závislých složek systému.

1.1 Použitá notace

V dalším textu budeme používat následující označení pro:

nepovinný výskyt X	[X]
libovolný počet výskytů X (včetně žádného)	[X]*
volitelný výskyt X či Y (implicitně X)	{X/Y}
současný stisk klávesy CTRL (CONTROL) a klávesy X	^X
povinný výskyt mezery	_

2 Technické vybavení osobních počítačů

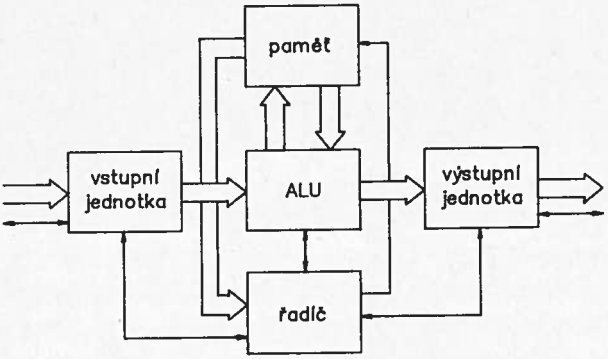
Operační systém počítače představuje základní vrstvu programového vybavení, která přímo spolupracuje s technickým vybavením. Pochopení principů činnosti a poslání operačního systému vyžaduje jistou úroveň znalostí technického vybavení. V této kapitole se proto věnujme přehledu technického vybavení osobních počítačů. Vzhledem k zaměření publikace a k rozmanitosti variant technického vybavení se soustředíme především na základní principy konstrukce 8bitových mikropočítačů. Podrobnější přehled technického vybavení osobních počítačů lze nalézt ve specializovaných publikacích, např. [8], [24], [27].

2.1 Prvky mikropočítačového výpočetního systému

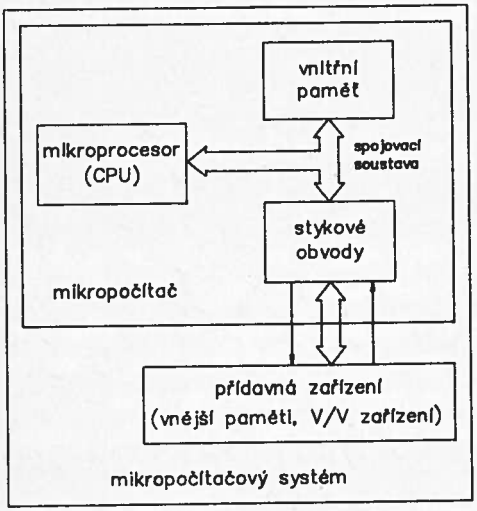
Běžný *mikropočítačový výpočetní systém* má klasickou strukturu programovatelného automatu von Neumannovy koncepce. Na obr. 2.1 je znázorněno klasické blokové schéma číslicového počítače von Neumannovy koncepce. Základní charakteristické rysy takového systému jsou zejména nezávislost počítače na typu řešené úlohy, jednotná architektura paměti, která slouží pro ukládání instrukcí programu i dat a sekvenční provádění programu dle instrukcí v paměti.

Typická struktura mikropočítačového výpočetního systému je uvedena na obr. 2.2. Mikropočítačový systém se skládá z *mikropočítače* a *přídavných zařízení*. Základními prvky mikropočítače jsou *procesor*, *paměť* pro ukládání programů a dat a prostředky pro ovládání přídavných zařízení, které nazýváme *stykové obvody*.

Procesor a paměť jsou potřebné pro vytvoření výpočetního procesu, stykové obvody umožňují komunikaci mikropočítače s vnějším prostředím. Jednotlivé prvky mikropočítače jsou propojeny *spojovací soustavou.*



Obr. 2.1. Blokové schéma číslicového počítače von Neumannovy koncepce



Obr. 2.2. Struktura mikropočítačového systému

Centrální jednotku mikropočítače představuje procesor (CPU – Central Processing Unit). Na rozdíl od jiných počítačů je procesor mikropočítače obvykle tvořen jediným integrovaným obvodem – *mikroprocesorem*. Odtud pochází označení mikropočítač. Technologický pokrok dovolil

v oblasti integrovaných obvodů zhustit potřebné množství součástek do jediného prvku a odrazil se i ve velikosti ostatních prvků mikropočítačového systému, kde však předponu mikro používáme výjimečně. Využití mikroprocesorů a ostatních obvodů vysoké integrace značně snižuje složitost návrhu mikropočítače. Poměrná jednoduchost konstrukce má ale jeden negativní důsledek. Vzhledem k dostupnému repertoáru součástek lze navrhnout velké množství různých, vzájemně neslučitelných mikropočítačů. Důvodem neslučitelnosti nemusí být jen použití jiných součástek, ale i způsob jejich propojení. Mikropočítač tudíž nelze charakterizovat jen typem použitého mikroprocesoru (tj. instrukčním repertoárem), ale i konfigurací paměti, spojovací soustavou a způsobem připojení přídavných zařízení. Uvedených možností bylo, je a pravděpodobně nadále bude bohatě využíváno.

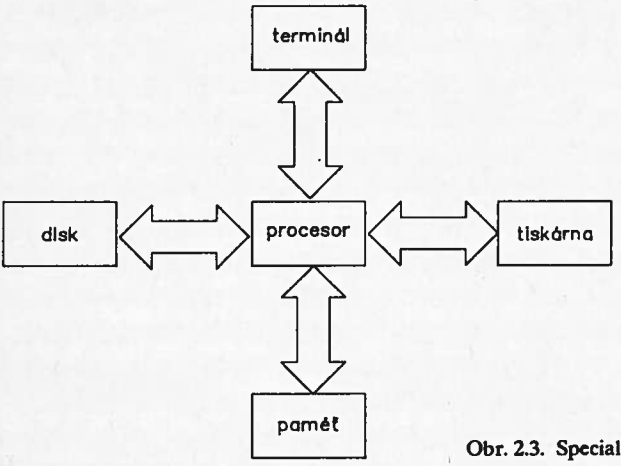
Důsledkem této rozmanitosti je nepřenosnost programového vybavení, neboť pro přenos programu z jednoho mikropočítače na druhý (byť osazených jedním typem mikroprocesoru) je nutno značnou část programu upravit. Tato situace začala být zvlášť nepříjemně pociťována po odeznění počátečního nadšení a byly hledány cesty ke standardizaci.

Standardizaci lze dosáhnout buď programově – vsunutím vrstvy mezi rozmanité technické vybavení a ostatní programy (např. u 8bitových mikropočítačů je touto vrstvou převážně operační systém CP/M), nebo unifikací technického vybavení (např. pro 16bitové mikropočítače díky firmě IBM a jejímu osobnímu počítači). Existují dokonce normy, předepisující určitý typ technického i programového vybavení, čímž lze dosáhnout standardizace uživatelského prostředí. Příkladem takové normy je norma MSX, propagovaná zejména v Japonsku.

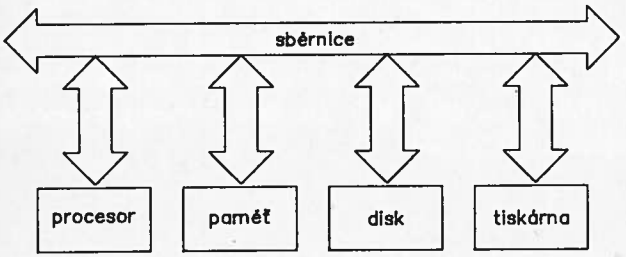
2.2 Sběrnice – spojovací soustava mikropočítače

Spojovací soustava počítače zajišťuje přenos informace mezi prvky systému. Různé architektury počítačů se často liší právě spojovací soustavou. Spojovací soustavu lze koncipovat jako sadu *speciálních propojovacích cest* pro každou dvojici prvků, mezi nimiž se informace přenáší. Takto specializovaná spojovací soustava (obr. 2.3) dovoluje maximální využití možností např. z hlediska rychlosti přenosu, je však poměrně ekonomicky náročná a málo pružná s ohledem na možné změny konfigurace prvků.

Opačným řešením je *univerzální spojovací soustava* (obr. 2.4), kdy se pro přenos informace mezi libovolnou dvojicí prvků používá jediné společné přenosové médium – *sběrnice* (angl. bus). Hlavní předností sběrnicové soustavy je celkové snížení počtu propojovacích linek (vodičů), což představuje nižší náklady a vyšší spolehlivost soustavy jako celku. Nevýhodou je složitější



Obr. 2.3. Specializovaná spojovací soustava



Obr. 2.4. Univerzální spojovací soustava

technika komunikace (přenosové médium je nutno sdílet) a nižší rychlost přenosu informace. U větších počítačů se obvykle používá kombinace obou technik. Pro mikropočítače je však použití sběrnice charakteristické.

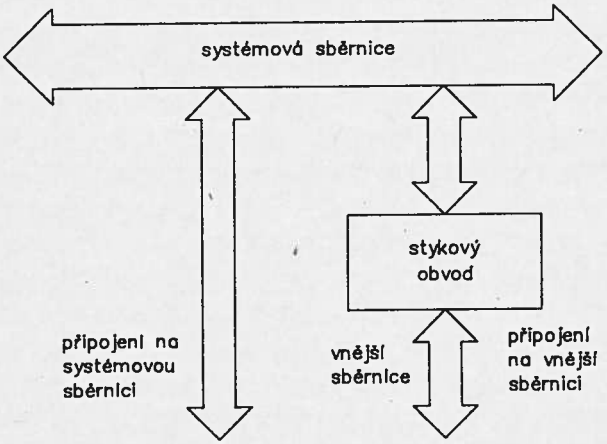
Sběrnice mohou být vytvářeny na několika úrovních. Sběrnici, která propojuje prvky mikropočítače, nazýváme *systémovou sběrnicí*. Konstrukční

provedení mikropočítače někdy vyžaduje rozdělení prvků mikropočítače do sady modulů. *Modulem* se rozumí samostatný konstrukční prvek (deska), který se připojuje na společnou systémovou sběrnici. Ve vnitřním řešení modulu lze opět využít sběrnice. Modul může mít např. na vnitřní sběrnici připojen procesor, základní paměť, příp. některá přídavná zařízení a stykové obvody pro připojení na systémovou sběrnici.

Pokud jsou všechny prvky mikropočítače umístěny do jediného modulu (tzv. jednodeskové mikropočítače), splývá systémová sběrnice se *sběrnicí modulu*. Jednoduché jednodeskové mikropočítače systémovou sběrnici vyvedenu nemají, příp. mají vyvedenu pouze část systémové sběrnice.

Vlastní jádro mikropočítače – mikroprocesor, bývá rovněž tvořen bloky, propojenými *interní sběrnicí*. Stejným způsobem bývají řešeny i další obvody vyšší integrace.

Rovněž pro *připojení přídavných zařízení* k mikropočítači lze využít sběrnice. Někdy je pro tento účel využito vyvedení systémové sběrnice nebo její části. Často je výhodnější využít pro rozšiřování konfigurace mikropočítačového systému speciální vnější sběrnici, kterou nazýváme *vnější rozhraní mikropočítače*. Obě možnosti ilustruje obr. 2.5. Stykové obvody mezi systémovou sběrnicí a vnější sběrnicí mohou odstínit speciality systémové sběrnice daného mikropočítače a naopak přizpůsobit vnější rozhraní doporučeným standardům. Přestože zařazení stykových obvodů zpomaluje



Obr. 2.5. Dvě možnosti rozšiřování konfigurace mikropočítače

komunikaci, má použití vnější sběrnice (s výjimkou extrémně rychlých přenosů) své nesporné výhody.

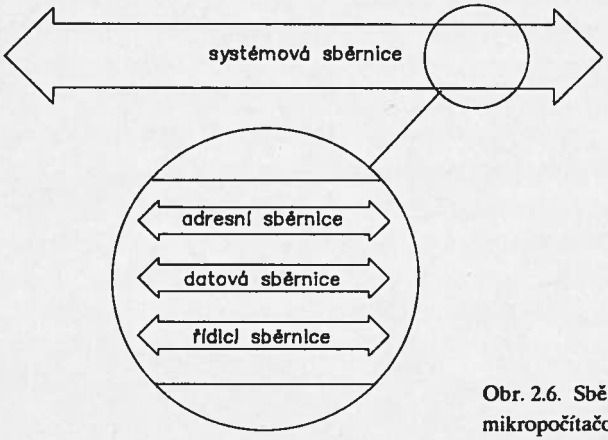
Důsledkem sběrnicové architektury mikropočítačových systémů je jejich poměrně snadná rozšiřitelnost. Mikropočítače s vyvedenou systémovou sběrnicí lze dodatečně rozšiřovat doplňkovými moduly, přičemž instalace modulu znamená většinou jen jeho zasunutí do konektoru. Mikropočítače, které mají vyvedeno vnější rozhraní, lze snadno doplňovat o nová zařízení, příp. připojovat na jiné počítače. Výhody sběrnicové architektury vyniknou zejména při dostatečné standardizaci sběrnice. Existují proto snahy o zavedení odpovídajících standardů. Již od prvních mikropočítačů (Altair, firma MITS 1976) je známa *systémová sběrnice S-100,* určená pro propojování prvků mikropočítače a používaná u menších systémů dodnes. Její označení vychází z použitého počtu vodičů. Jiným standardem je systémová sběrnice osobních počítačů firmy IBM (*I/O channel*), která dovoluje vytváření řady doplňkových modulů pro tyto počítače. U nejnovějších osobních počítačů (např. řada PS/2 firmy IBM) se dokonce využívá sdílení sběrnice pro několik přenosů současně (tzv. *microchannel*). *Řadič sběrnice* zajišťuje přepínání sběrnice v čase tak, aby se zvýšilo využití přenosového média.

Definice sběrnice shrnuje mechanickou, elektrickou i logickou stránku propojení. Systémová sběrnice bývá po mechanické stránce řešena paralelně propojenými konektory. Nedílnou součástí definice sběrnice je i způsob zapojení konektorů a popis elektrických vlastností signálů. Některé řídicí signály mohou být rozvedeny jinak (např. sériové propojení řetězce žádostí o přerušení).

Vodiče tvořící sběrnici lze rozdělit do dvou skupin – *vlastní informační vodiče* a *řídicí vodiče.* Řídicí vodiče slouží především pro ovládání správné funkce sběrnicové soustavy, tj. k zajištění správného přenosu po informačních vodičích. Charakteristická sběrnice mikropočítače má informační vodiče dále rozděleny na adresní a datovou část (obr. 2.6). Po *adresní sběrnici* se přenáší další identifikace informace (adresa v operační paměti, určení přídavného zařízení apod.). Vlastní přenos údajů probíhá po datové sběrnici.

Jednotlivé prvky mikropočítače (procesor, paměť, stykové obvody přídavných zařízení) jsou na sběrnici připojeny paralelně. Paralelní připojení integrovaných obvodů je umožněno tzv. *třístavovými výstupy prvků.* Třístavový výstup se může nacházet ve stavu logické nuly, logické jedničky nebo ve stavu vysoké impendance. V tomto stavu je v podstatě od sběrnice odpojen.

Prvek může být ke sběrnici připojen jednosměrně – pouze jako vstup, resp. jako výstup (přes třístavovou logiku), nebo obousměrně (jako vstup i výstup, v závislosti na řídicích signálech). Z paralelně připojených třístavových vývodů jsou samozřejmě v daném okamžiku aktivní pouze některé, ostatní jsou ve stavu vysoké impendance.



Obr. 2.6. Sběrnice mikropočítačového systému

Činnost na sběrnici musí být řízena, neboť je nutno zabezpečit sdílení společné přenosové cesty. Řízení sběrnice může být zajištěno centrálním dozorem, pomocí speciálního prvku nazývaného *řadič sběrnice*. Jiná koncepce využívá rozptýlené inteligence řízení činnosti na sběrnici, kdy je sdílení sběrnice dosahováno na základě vzájemné dohody mezi prvky, příp. může funkci řadiče přebírat více prvků.

Sdílení sběrnice při centrálním dozoru probíhá obvykle tak, že řadič sběrnice (tzv. *master*) registruje žádosti o přidělení sběrnice od ostatních prvků připojených na sběrnici (tzv. *slaves*). Na základě jistých kritérií (*priorit*) určí řadič jednoho z žadatelů (výstupní prvek) jako *vysílač* (*transmiter*). Ostatní výstupní prvky, které se komunikace nezúčastní, jsou ve stavu vysoké impedance. Vstupní prvky představují možné *posluchače* (*receivers*).

Po ukončení výběrové fáze je sběrnice přidělena vysílači. Vysílač zpravidla provede další selekci v posluchačích, např. vysláním identifikace žádaných posluchačů na adresní sběrnici. Výběrové obvody vstupních prvků na základě informací z řídicí a adresní sběrnice aktivují pouze žádané posluchače. Sběrnice je obsazena vysílačem až do ukončení přenosu, či příp.

přerušení přenosu na základě žádosti s vyšší prioritou. Výjimečně může být současně aktivních více výstupů, např. při použití společného vodiče pro všechny žádosti o přerušení. Obsluha podobných situací však vyžaduje další logiku pro rozhodnutí o aktivních výstupech a řešení jejich priority.

V mikropočítačových systémech přebírá velmi často funkci řadiče sběrnice přímo mikroprocesor. V některých řešeních mikropočítačů může mikroprocesor předat řízení sběrnice na určitou dobu jinému řadiči (např. obvodům přímého přístupu do paměti). Pro jednoduchou realizaci vzájemné dohody mikroprocesoru s jinými obvody bývá mikroprocesor vybaven vstupním signálem, kterým prezentujeme požadavek na dočasné odpojení mikroprocesoru od sběrnice systému. Po akceptování požadavku mikroprocesorem může funkci řadiče sběrnice převzít jiný prvek.

Sdílení sběrnice při rozptýlené inteligenci probíhá obdobně jako při centrálním dozoru, pouze funkci řadiče zastává pokaždé jiný prvek. Předá-li např. mikroprocesor funkci řadiče sběrnice obvodům pro přímý přístup do paměti, zajistí výběr vysílače a posluchačů tyto obvody – vysláním vhodných informací na řídicí a adresní sběrnici.

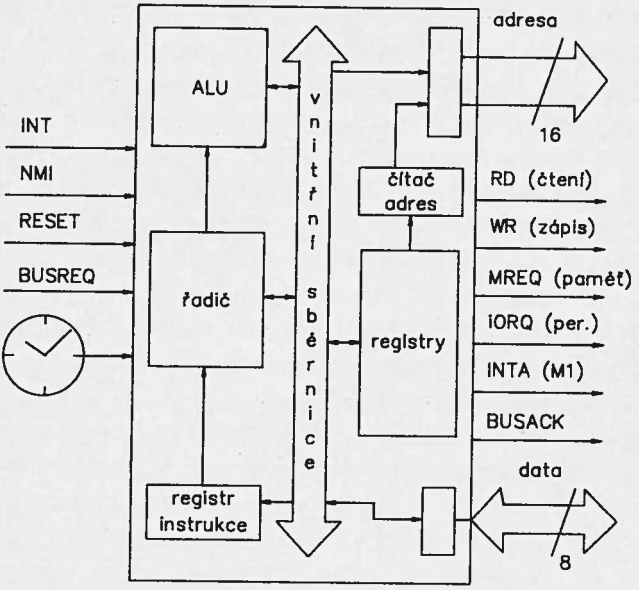
2.3 Mikroprocesor

Centrální jednotku mikropočítače tvoří *mikroprocesor.* Je to programovatelný automat, který je schopen přijímat a vykonávat přesně definovanou sadu povelů – strojových instrukcí. Posloupností těchto instrukcí je řízena celková činnost mikropočítače. Podrobný popis struktury a funkce různých mikroprocesorů se vymyká rámci této publikace. Lze jej nalézt např. v [8], [24], [27]. Zde uvedeme pouze informace potřebné pro pochopení dalšího textu. Jako vzor přitom použijeme typické představitele 8bitových mikroprocesorů, které tvoří základ technického vybavení pro mikropočítače vybavené operačním systémem CP/M.

2.3.1 Struktura a funkce mikroprocesoru

Vnitřní architektura mikroprocesoru obvykle také využívá vnitřní sběrnice, na níž jsou připojeny jednotlivé funkční bloky procesoru. Na obr. 2.7 je znázorněna (zjednodušená) struktura mikroprocesoru Z80, který

bývá nejčastěji využit v technickém vybavení pro systém CP/M. Samotný systém je distribuován v kódu mikroprocesoru Intel 8080, mikroprocesor Z80 je však schopen tento kód interpretovat. Většinu programů proto budeme prezentovat v kódu procesoru Z80. Srovnání instrukčních repertoárů obou procesorů je uvedeno v příloze.



Obr. 2.7. Struktura mikroprocesoru Z80

Základem mikroprocesoru je *řadič*, který zajišťuje spolupráci ostatních bloků mikroprocesoru při vykonávání instrukcí. Mikroprocesor zpracovává informaci, kódovanou pomocí čísel (přesněji pomocí bitových řetězců). Další důležitou součástí mikroprocesoru je proto *aritmeticko-logická jednotka* (ALU), jejímž úkolem je provádění matematických operací. Pro uložení argumentů a výsledků operací obsahuje mikroprocesor několik paměťových buněk – *registrů*.

Instrukce a data ke zpracování přijímá mikroprocesor z vnějšího prostředí a výsledky zpracování naopak do vnějšího prostředí předává. Pro tyto účely je vybaven *vyrovnávacím datovým registrem*, přes který je pomocí

sady datových vývodů propojen s vnějším prostředím – připojen na datovou sběrnici. Datové vývody mikroprocesoru jsou využívány pro vstup i výstup informace – jsou obousměrné.

Pro správný přenos dat mezi mikroprocesorem a ostatními prvky mikropočítače je nutno identifikovat přenášenou informaci. Identifikaci provádí mikroprocesor pomocí *adresy*. Adresou se rozlišují jak buňky operační paměti, tak i stykové obvody přídavných zařízení. Z toho důvodu obsahuje mikroprocesor vyrovávací *adresní registr*, který je sadou adresních vývodů připojen na adresní sběrnici. Adresní vývody jsou obvykle jednosměrné (výstupní) – adresu zadává mikroprocesor.

Datová sběrnice je mimo vstup a výstup dat využita i pro vstup instrukcí. Právě prováděná instrukce je uložena v *registru instrukce*. Pro návazné adresování instrukcí obsahuje mikroprocesor tzv. *čítač adres*, ukazující na instrukci, která se má provádět.

Počtem datových vývodů (šířkou datové sběrnice) je vymezen rozsah paralelně přenášených dat. Různé mikroprocesory se liší počtem datových vývodů, který představuje jednu ze základních charakteristik mikroprocesoru. První návrhy mikroprocesorů používaly 4 datové vývody (Intel 4004), dnešní mikroprocesory jich používají 8 (Intel 8080, 8085, Zilog Z80, Motorola 6800, Rockwell 6502), 16 (Intel 8086, Motorola 68000) nebo 32 (Intel 80386, Motorola 68030) datových vývodů, a tedy i odpovídající počet vodičů datové sběrnice. Podobně je šířkou adresové sběrnice omezen základní adresový prostor mikropočítače. V současné době se používá 16 (Intel 8080, 8085, Zilog Z80, Motorola 6800, Rockwell 6502), 20 (Intel 8086, 8088, 80186), 24 (Intel 80286, Motorola 68000, 68010) nebo 32 (Intel 80386, Motorola 68020, 68030) adresních vývodů (vodičů adresní sběrnice).

Některé mikroprocesory kvůli úspoře vývodů používají přepínání sběrnice v čase (časového multiplexu) buď na datové sběrnici (Intel 8088, 80188, Motorola 68008, 68000), nebo na adresní sběrnici (Motorola 68020, Intel 80286). Data či adresa se pak přenášejí jako několik po sobě následujících hodnot. Spolupracující obvody musí provádět příslušnou transformaci z rozděleného a na rozdělený tvar. Časový multiplex samozřejmě zpomaluje rychlost přenosu, neboť pro přenos jednoho údaje je zapotřebí několik přenosů po sběrnici.

Mimo datové a adresní vývody má mikroprocesor několik dalších vstupních a výstupních vývodů, které dovolují jeho ovládání či identifikace

stavu. O některých z nich je účelné se zmínit z hlediska dalšího výkladu. Pro označení probíraných vývodů použijeme zkratek používaných u řady Z80, u konkrétních mikroprocesorů se může použité značení mírně lišit.

Vstup RESET slouží pro nastavení mikroprocesoru do *počátečního stavu* – zejména je důležité nastavení pevné počáteční adresy do čítače adres. Na této adrese se musí nacházet v době startování mikropočítače tzv. *inicializační program.*

Dalším důležitým vstupem mikroprocesoru je vstup INT *žádosti o přerušení* (příp. několik vstupů žádostí s různou naléhavostí). Přerušení slouží pro efektivní synchronizaci činnosti mikroprocesoru s asynchronními procesy v okolí procesoru. Přivedení aktivního signálu na přerušovací vstup mikroprocesoru způsobí, že po dokončení rozpracované instrukce se prováděný proces přeruší a do čítače adres se nastaví nová adresa – adresa programu pro zpracování přerušení. Mikroprocesor současně nastaví výstupní signál INTA (INTerrupt Acknowledgement), kterým informuje spolupracující obvody o akceptování žádosti.

Asynchronní procesy v okolí mikroprocesoru mohou generovat novou žádost o přerušení ještě před ukončením zpracování žádosti předchozí. Aby bylo možno zabránit přerušení programu pro zpracování přerušení, musí být mikroprocesor vybaven tzv. *registrem povolení přerušení*, nazývaným *maska přerušení.* Obsah tohoto registru určuje, zda bude vstupní signál žádosti o přerušení akceptován či nikoliv. V okamžiku potvrzení žádosti o přerušení mikroprocesor automaticky zamaskuje vstup dalších přerušení. Masku přerušení lze programově ovládat, takže např. před ukončením zpracování přerušení odmaskujeme, čímž povolíme přijetí dalších přerušení.

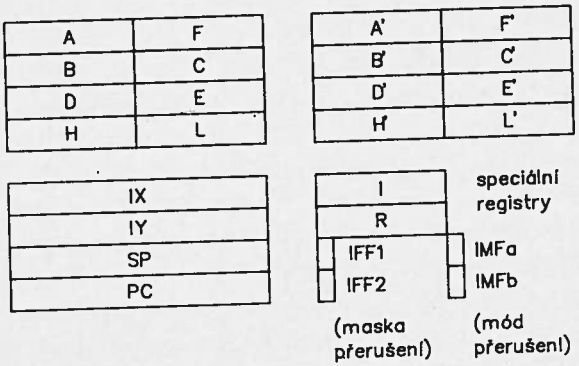
Mikroprocesor Z80 má dále speciální přerušovací vstup NMI (Non Maskable Interrupt), který nelze maskovat, sloužící pro řešení kritických situací (např. výpadek napájecího napětí). Podrobněji se rozborem zpracování přerušení budeme zabývat v odstavci o přerušovacím systému mikropočítače, kde se rovněž budeme zabývat otázkou určení adresy programu pro zpracování přerušení.

Přítomnost přerušovacího mechanismu bývá užita dvakrát. Proto lze obvykle vyvolat přerušení programově – v instrukčním repertoáru mikroprocesorů jsou instrukce, které způsobí stejný efekt jako vnější přerušení. Často je tento způsob aktivace podprogramů využit pro volání funkcí operačního systému. Některé mikroprocesory proto umožňují využít

přerušení pro změnu stavu mikroprocesoru – přechod z uživatelského do systémového režimu.

Většina mikroprocesorů obsahuje také dva speciální vývody, které umožňují odpojení mikroprocesoru od sběrnice v případech, kdy po sběrnici probíhá přenos mezi jinými prvky systému. Vstup BUSREQ (BUS REQuest) informuje mikroprocesor o vzniku požadavku na uvolnění systémové sběrnice. Mikroprocesor obvykle dokončí právě rozpracovaný přenos a odpojí se od sběrnice (adresní a datové vývody se převedou do stavu vysoké impedance). Spolupracující obvody informuje mikroprocesor o uvolnění sběrnice výstupním signálem BUSACK (BUS ACKnowledgement). Po vypnutí signálu BUSREQ pak mikroprocesor pokračuje v činnosti na sběrnici.

Směr pohybu dat po datové sběrnici určují řídicí signály. K jejich ovládání slouží výstupní vývody RD (Read Data – přenos dat směrem do mikroprocesoru) a WR (WRite data – přenos dat směrem z mikroprocesoru).



Obr. 2.8. Registry mikroprocesoru Z80

Některé mikroprocesory mají možnost odlišit případy, kdy chce mikroprocesor spolupracovat s operační pamětí a kdy s přídavným zařízením. Mikroprocesor má pak speciální instrukce pro spolupráci s pamětí a pro ovládání přídavných zařízení. Při provádění instrukce spolupracující s pamětí nastaví mikroprocesor výstupní signál MREQ (Memory REQuest), vstupní nebo výstupní instrukci indikuje signálem IORQ (Input/Output REQuest).

Činnost mikroprocesoru je řízena *hodinovým signálem*. Frekvence hodinových impulsů určuje rychlost provádění instrukcí a představuje další základní charakteristiku mikroprocesoru. Hodinový signál je rovněž základem synchronní spolupráce mikroprocesoru s ostatními prvky mikropočítače.

2.3.2 Programový model mikroprocesoru Z80

Na obr. 2.8 je znázorněna množina programově přístupných registrů mikroprocesoru Z80. Jejich obsah lze přímo nebo nepřímo měnit pomocí strojových instrukcí.

Registr A je využíván jako 8bitový střadač při většině aritmetických a logických operací. Registry B, C, D, E, H, L slouží jako univerzální registry pro uložení 8bitových údajů, nebo mohou být využívány jako dvojice BC, DE, HL pro uložení 16bitových údajů. V tom případě obsahují nižší bity údaje registry C, E nebo L.

Příznaky výsledků operací jsou uloženy ve stavovém registru F (Flags), který zahrnuje následující sadu příznaků:

S – příznak znaménka (S=0 – kladný výsledek),
Z – příznak nulového výsledku (Z=1 znamená A=0),
AC – příznak přenosu z 3. do 4. bitu,
P – příznak přetečení nebo parity,
N – příznak odčítání pro instrukci DAA,
C – příznak přenosu z nejvyššího bitu (Carry).

Kombinace střadače a stavového registru se nazývá *stavové slovo procesoru* a značí se AF.

Sada registrů A, B, C, D, E, F, H a L je v procesoru Z80 zdvojena, tj. existuje záložní sada registrů A´, B´, C´, D´, E´, F´, H´, L´, které slouží pro dočasné uložení stavu registrů A až L např. při rychlém přechodu na program pro obsluhu přerušení. Vzájemnou výměnu obsahu stavového slova AF a A´F´ lze provést jedinou instrukcí EX AF, A´F´. Podobně lze vyměnit obsah registrů B, C, D, E, H, L za obsah registrů B´, C´, D´, E´, H´ a L´ instrukcí EXX.

Programový model dále obsahuje 16bitové registry SP – *ukazatel na vrchol zásobníku* (Stack Pointer), PC – *čítač adres* (Program Counter) a registry IX a IY, které jsou využívány při indexovém způsobu adresace.

Sada speciálních registrů slouží zejména pro řízení zpracování přerušení. Mikroprocesor Z80 může zpracovávat maskovatelná přerušení různým způsobem podlę aktuálního módu zpracování přerušení, který je uložen v registrech IMFa a IMFb. Aktuální mód zpracování přerušení lze nastavit programově instrukcemi IM (IM 0, IM 1, IM 2).

V módu 0 je po akceptování signálu INT čtena následující instrukce z datové sběrnice, kam ji připraví přerušovací systém. Obvykle se jedná o instrukci RST, pomocí níž je možno provést rozskok na jednu z 8 adres dle zdroje přerušení. V módu 1 se na maskovatelné přerušení reaguje vždy voláním podprogramu na adrese 38H. Mód 2 umožňuje tzv. *vektorové zpracování přerušení.* Slabika získaná z datové sběrnice během identifikace zdroje přerušení se zkombinuje s obsahem registru I pro získání adresy programu pro obsluhu přerušení. 8bitový registr I tedy obsahuje adresu stránky v operační paměti, ve které se nachází aktuální tabulka vektorů adres programů pro obsluhu přerušení.

Jednobitový registr masky přerušení IFF1 slouží jako příznak povolení (IFF1=1) nebo blokování (IFF1=0) maskovatelných přerušení. Registr IFF2 slouží pro úschovu stavu registru IFF1 během zpracování nemaskovatelných přerušení.

Speciální registr R slouží jako čítač pro obnovu dynamických pamětí a programově se obvykle nevyužívá.

2.3.3 Instrukční repertoár mikroprocesoru Z80

Funkční schopnosti mikroprocesoru jsou vymezeny *repertoárem instrukcí.* Obecně lze v instrukčním repertoáru mikroprocesorů nalézt instrukce následujících skupin:

- instrukce pro nastavení a testování stavu mikroprocesoru (vnitřních datových, adresních a příznakových registrů, registru masky přerušení apod.),
- instrukce pro provádění aritmetických a logických operací,
- instrukce pro řízení postupu zpracování (skoky, volání podprogramů, programová přerušení a podmíněné verze těchto instrukcí),
- instrukce pro řízení vnější aktivity mikroprocesoru.

Množina instrukcí mikroprocesoru Z80 tvoří nadmnožinu instrukčního repertoáru mikroprocesoru Intel 8080. Instrukce lze rozdělit na následující skupiny:

– instrukce pro ovládání stavu mikroprocesoru,
– instrukce pro aritmetické a logické operace,
- instrukce pro přesuny,
– instrukce pro práci s bloky dat,
– instrukce pro práci s jednotlivými bity dat,
– instrukce pro řízení postupu zpracování,
– instrukce pro řízení vstupu a výstupu.

Přehled instrukcí mikroprocesoru Z80 je uveden v Příloze.

V programech uváděných dále v textu zapisujeme programy pro mikroprocesor Z80 v jazyce symbolických instrukcí – *asembleru*. Obecně má zápis instrukce v asembleru tvar:

[návěští [:]] [operace] [operandy] [; komentář].

Jako *návěští* je možno použít libovolný identifikátor (musí začínat písmenem) o délce 1 až 16 znaků. Malá a velká písmena mají stejný význam. Uvnitř návěští se může vyskytovat znak " $ ", který je ignorován. Za návěštím může a nemusí být uveden znak " : ". Jako návěští nelze použít *rezervovaná slova* – tj. symbolická označení instrukcí Z80, pseudoinstrukcí (direktiv), označení registrů programového modelu (A, B, C, D, E, H, L, F, AF, AF', BC, DE, HL, SP, IX, IY) nebo symbolické označení podmínky (NZ, Z, NC, C, PO, PE, P, M). Pokud je v zápisu instrukce uvedeno návěští, nabývá odpovídající identifikátor hodnoty aktuálního čítače adres.

Na místě *operace* lze použít libovolný symbol instrukce pro Z80 nebo pseudoinstrukce (direktivy). Přehled symbolických označení instrukcí je uveden v Příloze, přehled direktiv dále v textu současně se stručným popisem významu.

Operandy mohou být numerické konstanty, návěští nebo výrazy složené z numerických konstant (binárních, oktalových, dekadických či hexadecimálních), řetězů, identifikátorů, rezervovaných symbolů a formované pomocí operátorů +, -, *, /, mod, not, and, or, xor, shr, shl. Jejich priorita je dána:

1. *, /, mod, shr, shl
2. +, -
3. not
4. and
5. or, xor,

dále zleva doprava a lze ji změnit pomocí závorek.

Numerické konstanty zapisujeme v příslušné soustavě jako posloupnost číslic, za kterou bezprostředně následuje označení soustavy – tj. "B" (binární), "O", "Q" (oktalová), "H" (hexadecimální) nebo "D" (dekadická). Konstanta bez označení soustavy je chápána jako dekadická. Řetěz je posloupnost znaků uzavřená mezi znaky " ' ". Chceme-li v řetězu uvést znak " ' " , musíme jej zapsat jako dvojici " ' ' " . Řetěz může obsahovat maximálně 64 znaků, malá písmena se v tomto případě liší od velkých.

V symbolickém vyjádření instrukcí je dodržena zásada, aby pořadí operandů v instrukci připomínalo zápis přiřazovacího příkazu ve vyšším programovacím jazyce. Výstupní operand je proto uváděn jako první, např. zápis instrukce pro přesun hodnoty na adrese (která je obsahem registrového páru HL) do střadače, má tvar:

LD A,(HL).

Z příkladu je zřejmé, že závorkami je vyznačena nepřímá adresa. Instrukce LD A,66 způsobí přesun hodnoty 66 do střadače A, zatímco instrukce LD A,(66) přesune do střadače obsah slabiky na adrese 66.

Mimo vlastní symbolický zápis instrukcí používáme v zápisu programů následující *pseudoinstrukce* (*direktivy*), sloužící pro organizaci uložení programu a dat.

Direktiva ORG

[návěští] ORG výraz

Direktiva ORG nastaví čítač adres na hodnotu určenou výrazem (0...0FFFFH). Pokud je v direktivě ORG uvedeno návěští, nabývá hodnoty výrazu uvedeného za ORG.

Direktiva END

[návěští] END [výraz]

Pokud je uvedena v programu direktiva END, znamená logický konec programu. Pokud je uvedeno v direktivě END návěští, nabývá hodnoty aktuálního čítače adres. Výrazem za symbolem END je určena startovací adresa programu (obvykle 100H). Není-li výraz uveden, je startovací adresa 0.

Direktiva EQU

návěští EQU výraz

Direktiva EQU slouží pro trvalou definici hodnoty symbolu. Návěští před EQU nabývá hodnoty výrazu a může být definováno pouze jednou.

Direktiva SET

návěští SET výraz

Direktiva SET slouží pro definici hodnoty symbolu. Návěští před SET nabývá hodnoty výrazu a platí až do další direktivy SET. Slouží obvykle pro nastavení parametrů pro podmíněný překlad.

Direktiva IF

IF výraz
  příkazy
ENDIF

Podmíněný překlad: jestliže hodnota výrazu v okamžiku překladu IF je nula, nejsou další příkazy až do prvního výstupu ENDIF (včetně) překládány. Je-li hodnota výrazu nenulová, překlad se provádí.

Direktiva DB

[návěští] DB výraz [, výraz]*

Direktiva DB slouží pro definici obsahu paměti (Define Byte). Je-li uvedeno návěští, nabývá hodnoty čítače adres. Hodnotou výrazu (výrazů) je určen obsah paměti počínaje čítačem adres. Výrazy musí mít hodnotu v rozsahu 0...255, příp. řetěz znaků (maximálně 64 znaků). V jedné direktivě DB je možno zapsat jeden až maximálně 8 výrazů oddělených znakem " , ".

Direktiva DW

[návěští] DW výraz [, výraz] *

Direktiva DW (Define Word) má funkci obdobnou jako direktiva DB, ale vytváří celá slova. Výrazy musí mít hodnotu v rozsahu 0...65535, příp. jeden či dva znaky. Hodnota slova je ukládána do paměti v pořadí nižší slabika (výraz mod 256) – vyšší slabika (výraz div 256).

Direktiva DS

[návěští] DS výraz

Direktiva DS slouží k vymezení místa v paměti bez určení konkrétního obsahu. Pokud je v direktivě DS uvedeno návěští, nabývá hodnoty aktuálního čítače adres. Direktiva DS posune hodnotu čítače adres o hodnotu výrazu uvedeného za symbolem DS. Obsah paměti není definován.

2.4 Přerušovací systém mikropočítače

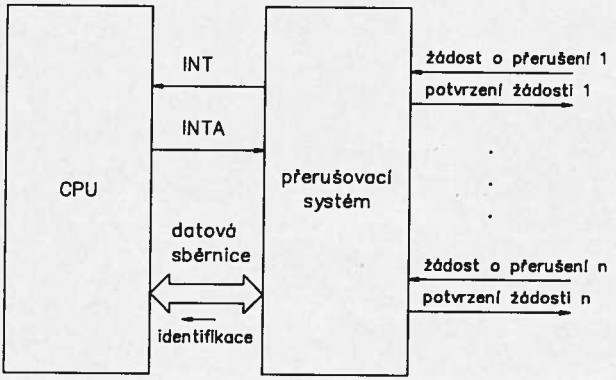
V každém výpočetním prostředí mohou probíhat do značné míry nezávislé procesy. Např. během doby, kdy tiskárna provádí tisk znaku, je její činnost nezávislá na ostatních komponentách počítače. Nezávislost ovšem skončí v okamžiku, kdy je tisk znaku dokončen a tiskárna je schopna přijmout další znak. Zde je nutno činnost tiskárny synchronizovat s procesem, který znaky pro tisk připravuje.

Obecně lze k synchronizaci nezávislých činností využít stavových informací a zabezpečit koordinaci *programově*. V takovém případě technické vybavení poskytuje pouze prostředky pro získání informací o stavu asynchronního procesu. Synchronizace se dosahuje programově *čekací smyčkou*, v níž se předaná stavová informace testuje a zpracovává. Podobně lze programově obsloužit i rozmanité výjimečné stavy a výskyt chyb.

Aktivní čekání v programových smyčkách nepředstavuje nejproduktivnější činnost systému. Mimoto může být reakce na vznik události značně zpožděna, neboť v programové čekací smyčce je nutno testovat výskyt všech možných událostí. Zjištění události je možné až v okamžiku testování stavu odpovídajícího prostředku. Určitou výhodou programových čekacích smyček je ryze sekvenční charakter zpracování. Na druhé straně však událost s velkou naléhavostí může čekat na dokončení zpracování byť zcela zanedbatelného problému. Řešení prioritního zpracování různě významných událostí je poměrně náročné a programově rozsáhlé. Proto je mnohem výhodnější přesunout realizaci čekacích smyček do technického vybavení pomocí *mechanismu přerušení*.

V odst. 2.3.1 jsme se zmínili o přerušovacím mechanismu zabudovaném v mikroprocesoru a aktivovaném přes speciální přerušovací vstupy (např. INT a NMI pro mikroprocesor Z80). Na jisté úrovni abstrakce si lze mechanismus přerušení představit jako speciální volání podprogramu, které je aktivováno buď vnější podmínkou – přerušením, přivedeným na odpovídající vstup mikroprocesoru, nebo speciální instrukcí.

Při využití přerušovacího mechanismu mikroprocesoru v rámci mikropočítačového systému vzniká zásadní problém. Zdrojů vnějších přerušení je v mikropočítačovém systému obvykle více, než kolik má mikroprocesor přerušovacích vstupů. *Přerušovací systém* mikropočítače proto zajišťuje zpracování signálů z různých obvodů mikropočítače a generuje signály přerušení pro mikroprocesor (obr. 2.9).

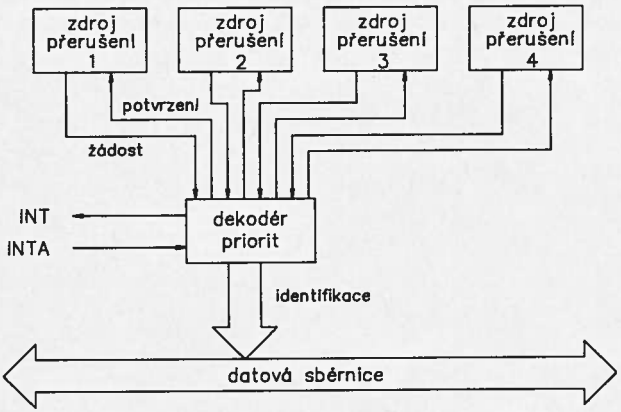


Obr. 2.9. Princip přerušovacího systému

Protože je možných zdrojů přerušení více, je nutno v okamžiku přijetí přerušení zjistit, která událost přerušení vyvolalą. Identifikaci je možno provést programově, tzn. přečtením příslušných stavových registrů. To je ovšem činnost časově náročná, a proto se častěji využívá pro identifikaci technických prostředků, zabudovaných do přerušovacího systému.

Technická realizace identifikace zdroje přerušení závisí na nárocích na schopnosti přerušovacího systému. V nejobecnějším případě jsou přivedeny na vstupy přerušovacího systému všechny žádosti o přerušení

(*paralelní řešení*). Každému zdroji přerušení je přiděleno jednoznačné identifikační číslo přerušení. Řadič přerušení vybere žádost s nejvyšší prioritou, připraví si její identifikační číslo a vyšle signál požadavku na přerušení INT do mikroprocesoru (obr. 2.10). V okamžiku, kdy mikroprocesor žádost o přerušení akceptuje (což hlásí signálem INTA – přerušení povoleno), vyšle řadič přerušení na datovou sběrnici identifikační číslo přerušení. Řadič mikroprocesoru toto číslo využije pro generování adresy obslužného programu v operační paměti.



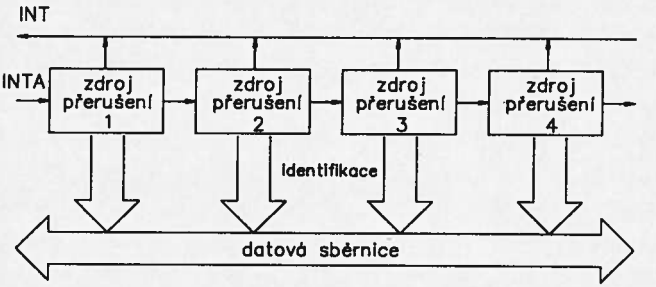
Obr. 2.10. Paralelní řešení přerušovacího systému

Často užívaná technika generování adresy vychází z představy, že identifikační číslo přerušení představuje index do tabulky adres (tzv. přerušovacího vektoru) obslužných podprogramů. Z ní pak řadič procesoru vybere vlastní adresu obslužného podprogramu, kterou se nahraje do čítače adres.

V jednodušších případech, kdy chceme ušetřit počet vodičů na řídicí sběrnici, lze použít tzv. *sériového zpracování žádosti o přerušení*, naznačeného na obr. 2.11. Ke každému zdroji přerušení přidáme navíc jeden vstup a jeden výstup. Vstup bude mít význam povolení žádosti o přerušení. Pokud obvod o přerušení nežádal, přenese povolení pouze na nový výstup. Žádal-li však o přerušení, nepřepustí povolení na výstup, a místo toho vyšle na datovou sběrnici své identifikační číslo. Jednotlivé zdroje přerušení jsou

propojeny sérivě vždy výstup obvodu s vyšší prioritou na vstup obvodu s nižší prioritou. Na vstup obvodu s nejvyšší prioritou je připojen signál povolení přerušení od procesoru.

Na rozdíl od paralelního řešení přerušovacího systému postačí v sériovém řešení jediný vodič řetězce přerušení, pokud nechceme rozlišovat různé úrovně přerušení. Přijde-li totiž žádost o přerušení po společném vodiči, nevíme který konkrétní prvek z řetězu o přerušení žádal. Pak ale také neznáme naléhavost příčiny přerušení, kterou bychom zjistili až po



Obr. 2.11. Sériové řešení přerušovacího systému

identifikaci zdroje přes potvrzení žádosti. Identifikaci může provést přerušovací systém, ale v případě, že zjištěná úroveň žádosti nemá na přerušení nárok, musí pak přerušovací systém udržovat frontu takto registrovaných žádostí. Realizaci víceúrovňového přerušovacího systému lze zajistit samostatným sériovým řetězcem pro každou úroveň, potom okamžitě víme, zda máme či nemáme žádost o přerušení procesoru předložit.

Návrh adekvátního přerušovacího systému představuje jeden z důležitých momentů při návrhu mikropočítače. Obvykle je výsledek určitým kompromisem naznačených možností. Je důležité si uvědomit, že využitím přerušovacího systému zavádíme do programového vybavení možnost paralelního zpracování. Obslužný program pro přerušení může např. přepsat buňku paměti, kterou využívá přerušený proces. Činnost systému je pak závislá na vzájemné časové poloze vzniku přerušení a stavu přerušeného procesu. Obvykle tyto problémy řeší operační systém – na úrovni aplikačních úloh lze pracovat opět sekvenčně.

2.5 Paměťový subsystém mikropočítače

K vytvoření výpočetního procesu je kromě procesoru nutná paměť, v níž jsou uložena data a program pro jejich zpracování. Principiálně se paměť počítače skládá ze sady buněk – *paměťových míst*. Ve von Neumannově modelu počítače jsou stejné buňky použity jak pro data, tak pro instrukce programu. Každá buňka paměti je jednoznačně identifikována číslem – *adresou* .

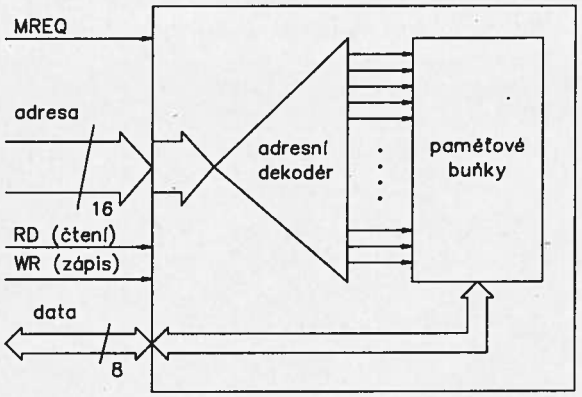
Počet paměťových míst představuje jednu ze základních charakteristik výpočetních schopností počítače – limituje totiž rozsah zpracovávaných dat i možnou velikost programu. Neméně důležitá je rychlost přístupu k buňkám, neboť ta ovlivňuje rychost zpracování. Přesněji řečeno, důležitý je tzv. *paměťový cyklus*, který se skládá z doby přístupu k buňce (čas potřebný pro její vyhledání podle adresy) a z doby potřebné pro přenos obsahu buňky (na sběrnici nebo ze sběrnice).

Z ekonomických i technických důvodů mají paměťové subsystémy počítačů hierarchickou strukturu. Je nutno hledat kompromis mezi kapacitou paměti a rychlostí přístupu na straně jedné a cenou na straně druhé. Paměťový subsystém se proto obvykle skládá nejméně ze dvou typů paměti: *vnitřní paměti* počítače s velkou rychlostí přístupu (řádově 10^{-7} až 10^{-6} sekund) a poměrně malou kapacitou (řádově 10^4 až 10^7 buněk) a z *vnějších pamětí* s menší rychlostí přístupu (řádově 10^{-4} až 10^2 sekund) a velkou kapacitou (řádově 10^4 až 10^{10} buněk). Vnější paměti mají mimoto důležitou funkci jako médium pro úschovu a přenos informace. Budeme se jimi podrobněji zabývat v odstavci o přídavných zařízeních (2.8.3). Procesor počítače operuje nad vnitřní pamětí – budeme ji proto nazývat *operační paměť*.

2.5.1 Operační paměť mikropočítače

Operační paměť mikropočítače je konstruována jako sada stejně velkých paměťových buněk. Velikost paměťové buňky je zpravidla určena šířkou datové sběrnice mikropočítače. Každá paměťová buňka je jednoznačně označena identifikačním číslem – *adresou*. Pomocí této adresy se na buňku odvolává mikroprocesor i ostatní prvky mikropočítače. Maximální počet paměťových buněk je obvykle určen šířkou adresní sběrnice systému.

Přístup k buňce paměti podle adresy zajišťuje *řadič paměti* za pomoci *adresního dekodéru*. Kromě adresy musíme řadiči paměti též sdělit, zda budeme z buňky číst, či do buňky zapisovat. Principiální schéma operační paměti je znázorněno na obr. 2.12, kde uvedené číselné údaje odpovídají typickému případu 8bitového systému.



Obr. 2.12. Principiální schéma operační paměti

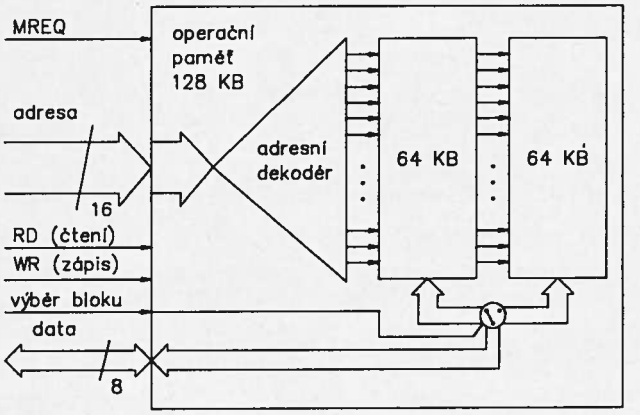
Konstrukční jednotkou operační paměti jsou zpravidla dvoustavové paměťové prvky, schopné zapamatovat si *bit* informace. Typickou minimální paměťovou buňkou je buňka o rozsahu 8 bitů (tzv. *slabika*, angl. *byte*), do níž lze uložit celkem 256 různých kombinací nul a jedniček. Kapacitu paměti budeme vyjadřovat buď v bitech (označíme b), nebo v násobcích slabik (značíme B, kde 1 B = 8 b). Přitom je výhodné používat binární násobky namísto dekadických, a proto používáme speciálního značení:

1 KB = 1 024 B (tj. 2^{10} B).
1 MB = 1 024 KB = 1 048 576 B, atd.

Někdy se s pamětí komunikuje po větších jednotkách, např. mluvíme o *slovech* (*word*), kde jedno slovo představuje 16bitovou buňku, či *dvojnásobných slovech* (*double word*) délky 32 bitů.

Z důvodů úspory vývodů mikroprocesoru nebo úspory vodičů sběrnice je možno adresu nebo data přenášet po částech, s využitím přepínání sběrnice v čase. Logická sběrnice je pak zásluhou doplňujícího technického

vybavení širší než skutečná fyzická sběrnice, samozřejmě na úkor rychlosti přenosu. Někdy se též využívá různých triků pro zvětšení adresního prostoru nebo velikosti paměťových buňek. Paměť lze např. sestavit z několika bloků, které lze různě připojovat na datovou sběrnici (tzv. *banked memory*). Jednotlivé bloky lze jednak přepínat pomocí přepínače, který ovládáme jako speciální vnější zařízení (tím dosáhneme zvětšení adresního prostoru), nebo je možné připojit různé bloky paměti na různé úseky datové sběrnice (tím se zvětší velikost paměťových buněk). Příklad podobného řešení ilustruje obr. 2.13 pro případ realizace paměti o kapacitě 128 KB z bloků 64 KB.



Obr. 2.13. Realizace paměti 128 KB z bloků 64 KB

K jednotlivým buňkám operační paměti můžeme přistoupit v náhodném pořadí dle zadané adresy. Proto bývá též operační paměť označována jako *paměť s libovolným přístupem* – RAM (Random Access Memory).

Operační paměť lze vytvořit z rozmanitých prvků, s využitím různých fyzikálních principů. Pokud lze informaci z operační paměti jak číst, tak i do paměti zapisovat, značíme ji dále jako paměť RWM (Read/Write Memory). Operační paměť bývá v mikropočítači obvykle realizována polovodičovými součástkami, neboť tak lze dosáhnout poměrně krátkého paměťového cyklu. Informace uložená v polovodičové paměti se však obvykle po odpojení napájecího napětí ztratí. Některé typy polovodičových pamětí dokonce

vyžadují pravidelnou obnovu uložené informace (tzv. *dynamické paměti*), jiné ji dokáží udržet, pokud jsou pod napětím (tzv. *statické paměti*).

Z určitých důvodů je nutné a výhodné používat i pamětí, jejichž obsah zůstává trvalý i po odpojení od napájení – tzv. *neporušitelná* (non-volatile) paměť. Její realizace je však cenově náročnější. Jednodušší je realizace pamětí, jejichž obsah je pevně nastaven a nelze jej během činnosti mikropočítače měnit. Takovou paměť nazýváme *pevná paměť* a lze v ní uchovávat programy nebo data, která jsou k dispozici okamžitě po zapnutí mikropočítače a která mohou například uvést mikropočítač do správného počátečního stavu. Protože lze z pevné paměti informaci pouze číst a nikoliv přepisovat, označujeme ji jako paměť typu ROM (Read Only Memory).

Operační paměť mikropočítače se tedy skládá z paměti, jejíž obsah lze měnit (typu RAM–RWM) a z pevné paměti (typu RAM–ROM). Někdy se používá zjednodušeného značení RAM nebo RWM pro paměť typu RAM–RWM a ROM pro paměť typu RAM–ROM, přestože pevná část operační paměti umožňuje rovněž libovolný přístup. Pokud nemůže dojít k nedorozumění, přidržíme se v dalším textu značení RWM a ROM.

Specializované mikropočítače pro řešení určité aplikace mohou mít většinu programů připravenou předem v pevné paměti. Pro univerzální systémy je naopak charakteristická změna funkce výměnou programu v paměti. Přesto i zde bývá určitá část programového vybavení umístěna v pevných pamětech (např. mikropočítač MacIntosh firmy Apple obsahuje až 200 KB pevné paměti se základním programovým vybavením).

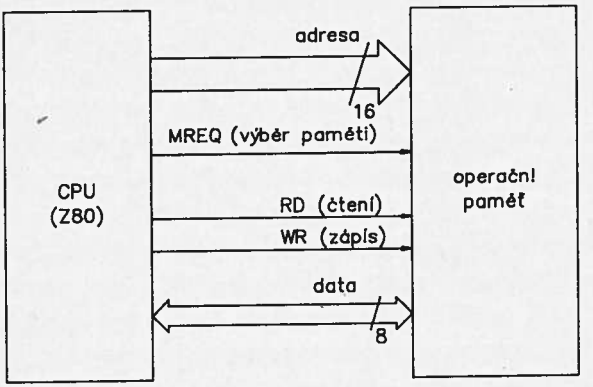
Obsah pevných pamětí bývá označován jako tzv. *firmware*, neboť jej obvykle připravuje výrobní firma. Do jisté míry je sporné, zda obsah pevných pamětí patří do technického vybavení (*hardware*), tj. do neměnitelných složek výpočetního systému, nebo do programového vybavení (*software*), tj. složek systému, které lze během činnosti měnit. U domácích mikropočítačů bývá někdy možná rychlá výměna části pevné paměti bez zásahu do mikropočítače (tzv. *software cartridges*).

2.5.2 Spolupráce mikroprocesoru s operační pamětí

Spolupráce mikroprocesoru s operační pamětí probíhá tak, že mikroprocesor požádá řadič sběrnice o přidělení sběrnice (není-li řadičem sběrnice sám). Po přidělení sběrnice zašle na adresní sběrnici požadovanou

adresu a nastaví řídicí signál RD (čtení) nebo WR (zápis). Řadič paměti na základě obdržené adresy zajistí přístup ke správné buňce paměti a přes datovou sběrnici přijme nebo vyšle odpovídající data.

Chybný požadavek, např. požadavek na zápis do pevné paměti nebo čtení či zápis do neexistující paměti (neobsazené části adresního prostoru), může vyvolat různou odezvu. U jednoduchých mikropočítačů obvykle takový požadavek projde na úrovni technického vybavení bez odezvy, náročnější systémy většinou generují přerušení.

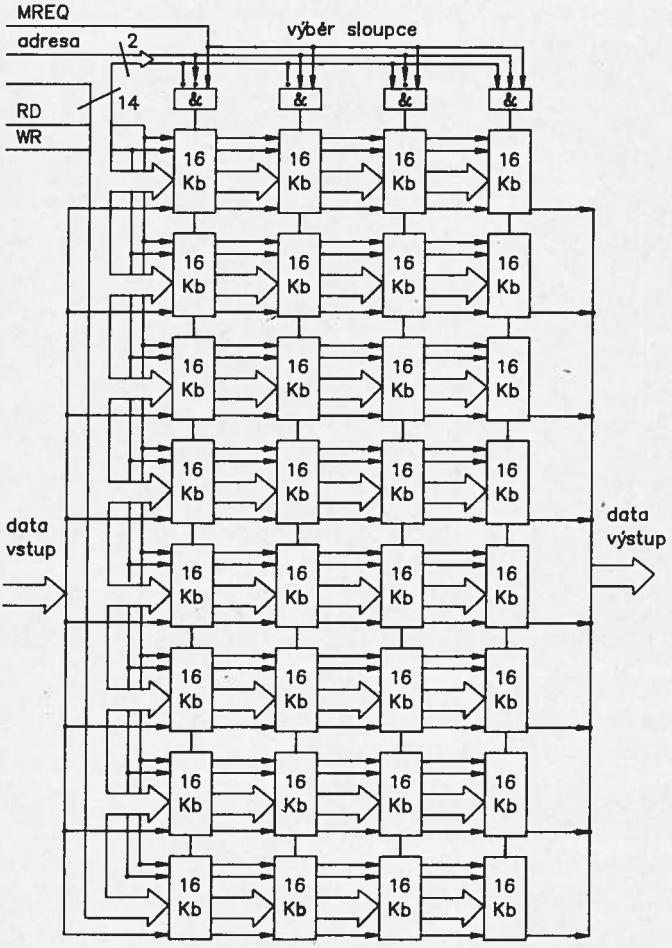


Obr. 2.14. Spolupráce mikroprocesoru s operační pamětí

U mikroprocesorů, které mají speciální instrukce pro spolupráci s vnějšími zařízeními (např. instrukce IN, OUT mikroprocesoru 8080), je v propojení využit řídicí signál MREQ, který slouží pro otevření cesty mezi mikroprocesorem a operační pamětí (viz obr. 2.14). Od jeho aktivace poslouchají pouze obvody operační paměti. Adresní i datovou sběrnici je totiž možno využívat i pro jiné účely, např. pro spolupráci s přídavným zařízením. Mikroprocesory, které nemají speciální instrukce pro vstup a výstup (např. mikroprocesory firmy Motorola), negenerují signál MREQ a spolupracují s přídavnými zařízeními shodně jako s operační pamětí. Selekci mezi posluchači na sběrnici je pak nutno provést výhradně na základě obsahu adresní sběrnice, odpadá předvýběr signálem MREQ.

Pro usnadnění výběru bývají konstrukční prvky (obvody) vybaveny speciálním vstupem CS (Chip Select). Konstrukční prvek je aktivní pouze

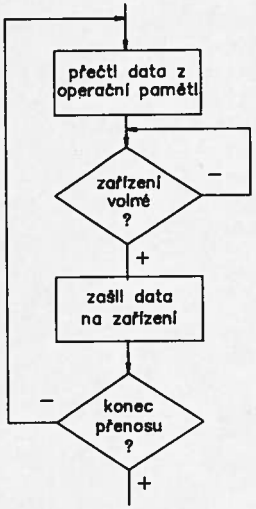
v případě aktivace vstupu CS. Příklad možné technické realizace operační paměti o rozsahu 64 KB z paměťových obvodů s kapacitou 16 Kb je uveden na obr. 2.15. Nejvyšší dva bity 16bitové adresy jsou využity k selekci jednoho ze čtyř sloupců. Selekce je dále podmíněna aktivací signálu MREQ, tj. požadavkem na spolupráci s pamětí. V každém sloupci je 8 obvodů, z nichž každý pamatuje 1 bit vybraný na základě zbylých 14 bitů adresy.



Obr. 2.15 Příklad technické realizace dekódování adresy

2.5.3 Přímý přístup do paměti

Sběrnicová architektura mikropočítače dovoluje, aby přenos po sběrnici probíhal bez zásahu mikroprocesoru. To je zvláště výhodné v případě přenosů mezi přídavným zařízením a operační pamětí. Pokud by nebylo možno z tohoto přenosu vyloučit mikroprocesor, musel by přenos probíhat následovně: mikroprocesor by přečetl data z operační paměti do vnitřního registru a poté je vyslal na přídavné zařízení, nebo obráceně. Na obr. 2.16 je



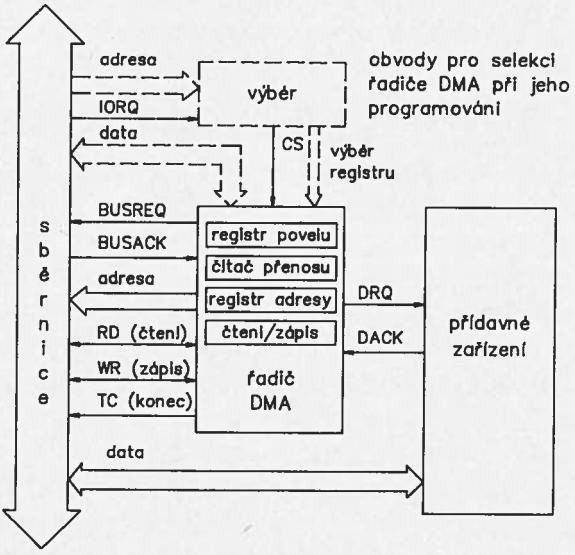
Obr. 2.16 Diagram řízení přenosu z operační paměti na přídavné zařízení

naznačen odpovídající postup činnosti mikroprocesoru. Je třeba si uvědomit, že mimo vlastní data je nutno po datové sběrnici přenášet i instrukce programu řídícího přenos, a to dříve než je mikroprocesor provede. Pro mikroprocesor Z80 lze např. použít následující fragment programu:

Operační kód	Symbolický tvar	Význam
21 xx xx	LD HL, adresa	; počáteční adresa přenosu
06 xx	LD B, délka	; délka přenosu
0E xx	LD C, port	; výstupní adresa
ED B3	OTIR	; přesun z paměti na výstup

V každém cyklu potřebujeme pro přenos jedné buňky z paměti na výstup přenést 2 buňky z paměti do mikroprocesoru (instrukci OTIR). V případě rychlého zařízení může tato režie zdržovat nebo dokonce vylučovat spolupráci. Výhodnější je pověřit přenosem speciální obvody, které mají uvedený program pevně zabudován a nevyžadují jeho čtení z operační paměti. Přenos pak probíhá bez zásahu mikroprocesoru přímým stykem s operační pamětí. Příslušné obvody proto nazýváme *řadič přímého přístupu do paměti*, zkráceně řadič DMA (Direct Memory Access).

Řadič přímého přístupu do paměti zprostředkuje přenos bloku dat, který samostatně rozdělí na řadu dílčích přenosů po jednotlivých informačních jednotkách (dle šířky datové sběrnice). Principiálně proto obsahuje 2 registry – *registr adresy přenášených dat* a *čítač*, obsahující počet ještě nepřenesených buněk. Další registr určuje směr přenosu – z operační paměti nebo do operační paměti. Všechny tyto registry představují řídicí registry obvodu a je nutno je nastavit před zahájením přenosu. Nastavení řídicích registrů nazýváme někdy *programování řadiče DMA*. U chytřejších řadičů lze nastavit i další chrakteristiky.



Obr. 2.17. Principiální schéma řadiče pro přímý přístup do paměti

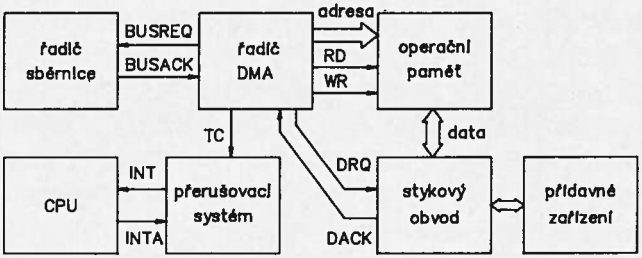
S okolním prostředím je řadič DMA propojen několika signály (obr. 2.17). Především jsou zde signály pro komunikaci s přídavným zařízením – *vstupní signál* DRQ (Data ReQuest), kterým přídavné zařízení hlásí, že je připraveno pro další dílčí přenos (přenos jedné informační jednotky) a *výstupní signál* DACK (Data reguest ACKnowledge), kterým řadič DMA potvrzuje, že dílčí přenos lze uskutečnit. Protože přenos probíhá po společné sběrnici, jsou zde dále nutné signály BUSREQ a BUSACK, kterými řadič DMA *žádá o přidělení sběrnice* (BUSREQ) a *dostavává potvrzení o jejím přidělení* (BUSACK). Obsah vnitřního registru, který udává směr přenosu, je podkladem pro generování *řídicích signálů* RD (čtení z paměti) a WR (zápis do paměti). Ukončení přenosu celého bloku dat je možno zjistit čtením stavového registru řadiče, nebo pomocí *výstupního signálu* TC (Terminal Count). Zahájení přenosu se obvykle provádí zápisem povelu do povelového registru řadiče DMA.

Mimo uvedené signály je z řadiče DMA vyveden obsah registru adresy dat a připojen na adresní sběrnici. Aby bylo možno řadič DMA programovat a příp. programově testovat jeho stav, je řadič připojen rovněž na datovou sběrnici. V době přenosu přímým přístupem je však obvykle od datové sběrnice odpojen. Výjimku tvoří řadiče DMA, které dovolují přesuny bloků dat v rámci operační paměti, kdy dílčí přenosy probíhají (ve dvou fázích) přes vnitřní datový registr obvodu DMA.

Spolupráce programu s řadičem DMA probíhá ve dvou fázích. V prvé fázi nastavíme programově registry řadiče přes datovou sběrnici systému, dle požadovaných parametrů přenosu. Po nastavení registrů řadiče je možno aktivovat druhou fázi zápisem povelu pro spuštění přenosu do povelového registru řadiče DMA. Ve druhé fázi pak probíhá přenos dat autonomně pod řízením řadiče DMA.

Řadič DMA v prvé řadě vyčká, až stykové obvody přídavného zařízení ohlásí, že jsou připraveny k zahájení dílčího přenosu signálem DRQ (viz obr. 2.18). Po příchodu signálu DRQ vydá řadič DMA požadavek na přidělení sběrnice aktivací signálu BUSREQ. Řadič sběrnice ve vhodném okamžiku žádost potvrdí signálem BUSACK, což způsobí připojení registru adresy řadiče DMA na adresní sběrnici. Je-li to v daném systému nutné, požádá řadič současně o přístup do operační paměti

signálem MREQ. Na jeho základě je pak sběrnice připojena též k operační paměti. Není-li signál MREQ v daném systému použit, postačí adresa, přivedená na adresní sběrnici.



Obr. 2.18 Principiální schéma systému s řadičem DMA

Při přenosu z operační paměti na výstup požádá řadič DMA signálem RD o čtení, tj. přenesení obsahu adresované buňky na datovou sběrnici. Poté je nastaven signál DACK, který způsobí připojení datové sběrnice k datovému registru vybraného stykového obvodu výstupního zařízení, kam se zapíše obsah datové sběrnice. Tím je ukončen jeden dílčí přenos, neboť další zpracování obsahu datového registru (např. přenos do výstupního zařízení) je záležitostí stykového obvodu. V tomto okamžiku může řadič DMA uvolnit systémovou sběrnici zrušením signálu BUSREQ, neboť musí vyčkat příchodu signálu DRQ od stykového obvodu. Mezitím řadič DMA zvýší obsah vnitřního registru adresy a současně poznamená v čítači, že byl proveden další dílčí přenos. Pokud se tím čítač vynuluje, přenos celého bloku skončil, což je indikováno výstupním signálem TC (konec přenosu). Ten může být využit například pro generování přerušení stávajícího programu a obsloužení ukončení přenosu. Ukončení celého přenosu řadičem DMA lze zajistit rovněž programově, přečtením stavového registru řadiče DMA.

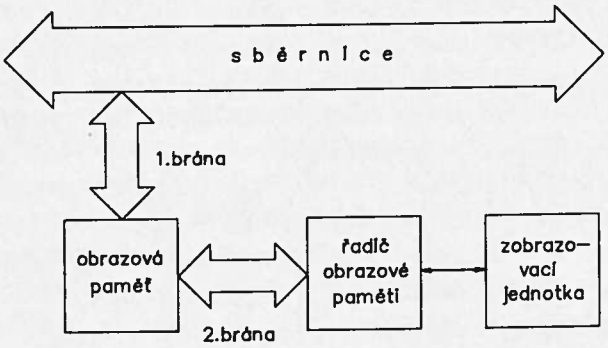
Přenos ze vstupu do operační paměti probíhá obdobně. Poznamenejme, že sběrnice nemusí být přidělena obvodu pro přímý přístup trvale po celou dobu přenosu – tím by se zcela zablokovala činnost ostatních prvků systému včetně mikroprocesoru, neboť alespoň instrukce, které má mikroprocesor provádět, je nutno po sběrnici. přenášet. Častěji se proto obvody přímého přístupu střídají s ostatními prvky systému. Pokud je mikroprocesor sám řadičem sběrnice, pak jednoduchou realizaci umožňují vývody

BUSREQ a BUSACK mikroprocesoru. Postačí propojit signály BUSREQ a BUSACK. Akceptování signálu na vstupu BUSREQ mikroprocesoru zajistí odpojení mikroprocesoru od sběrnice a generování potvrzení BUSACK.

Jinou možností je pozastavení činnosti mikroprocesoru zablokováním hodinového signálu (mluvíme o tzv. *kradení cyklů*). Mikroprocesor však musí být schopen udržet beze změny svůj vnitřní stav, což v případě použití dynamických pamětí pro vnitřní registry mikroprocesoru omezuje dobu pozastavení činnosti.

Obě uvedené metody zpomalují činnost procesoru. Pro náročnější systémy lze využít přísně synchronní spolupráci mikroprocesoru a řadiče DMA, kdy obvody DMA využívají sběrnici pouze v takových časových úsecích, kdy mikroprocesor sběrnici zaručeně nepoužívá, tzn. během autonomní činnosti procesoru na zpracování instrukce (dekódování instrukce, provádění matematických operací apod.). V tomto případě pracují oba prvky prakticky paralelně a nedochází ke zpomalení procesoru. Pokud ani toto řešení nepostačuje, je obvykle třeba upustit od jediné sběrnice systému, neboť nároky na přenos vyžadují zvýšení propustnosti, které již nelze zajistit.

U velmi náročných systémů může probíhat činnost obou složek zcela nezávisle (paralelně), například s využitím zdvojeného přístupu do paměti (tzv. *dvoubránová paměť*). Jako příklad lze uvést řešení grafické obrazové paměti náročnějších pracovních stanic (obr. 2.19), ke které má přístup nejen mikroprocesor přes systémovou sběrnici, ale rovněž grafický obrazový procesor po vlastní sběrnici s využitím DMA.



Obr. 2.19 Dvoubránová grafická obrazová paměť

Jednoduché obvody přímého přístupu je nutno pro každý další přenos znovu nastavit (naprogramovat). Chytřejší řadiče dovolují připravit předem parametry následujícího přenosu v záložních registrech. Vynulování čítače pak způsobí přehrání záložních registrů do řídicích registrů obvodu (tzv. autoload), čímž je automaticky aktivován nový přenos. Mikropočítačový systém se pak vůbec nemusí zabývat opakovanými přenosy. Často se tohoto způsobu využívá při údržbě obsahu dynamických pamětí, příp. při obnovování obrazu na obrazovce zobrazovací jednotky.

Mikropočítačový systém může obsahovat více řadičů pro přímý přístup do paměti, spolupracuje-li s více zařízeními. Jednodušší řadič DMA vytváří jednu přenosovou cestu mezi operační pamětí a stykovými obvody přídavného zařízení – tzv. kanál pro přímý přístup. Mocnější řadič DMA může obsluhovat více zařízení. Vytváří pak několik nezávislých kanálů a musí proto obsahovat sadu registrů pro každý kanál a řešit otázky priority kanálů, dojde-li současně k několika požadavkům na přenos. Pokud se nepředpokládá současný přenos po několika kanálech, lze využít přepínání jediného kanálu. Řadič pak musí obsahovat i registr adresy zařízení, se kterým má právě komunikovat.

2.6 Stykové obvody přídavných zařízení

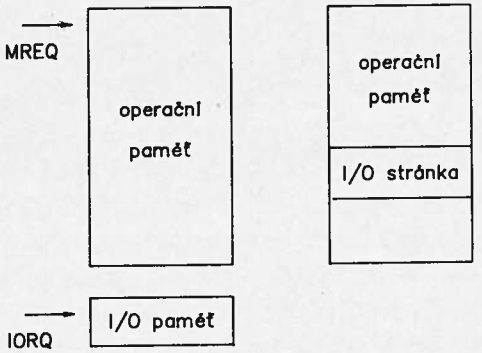
Spolupráce mikroprocesorového systému s vnějším prostředím je zprostředkována pomocí přídavných zařízení. Přídavná zařízení slouží pro vstup a výstup informace. Přenos informace řídí speciální *stykové obvody*. K nim je na jedné straně připojeno fyzické zařízení (např. tiskárna, klávesnice apod.) a na druhé straně jsou stykové obvody připojeny na systémovou sběrnici mikropočítače. Zhruba řečeno provádějí stykové obvody transformaci vnitřní formy informace v rámci mikropočítače do formy, kterou vyžaduje příslušné přídavné zařízení.

2.6.1 Struktura stykových obvodů

Potřeba stykových obvodů vyplývá z několika podmínek. Především je nutné elektrické přizpůsobení signálů mezi úrovněmi na straně mikropočítače (sběrnice) a vlastním přídavným zařízením. Neméně důležité je logické

přizpůsobení signálovým požadavkům zařízení. Bez zavedení stykových obvodů by řídicí sběrnice systému musela obsahovat všechny řídicí signály rozmanitých zařízení. Generování a zpracování řídicích signálů by bylo nutno zajistit programově – použití stykových obvodů ulehčuje práci mikroprocesoru. V neposlední řadě dovolují stykové obvody vyrovnání různých rychlostí zpracování.

Na určité úrovni abstrakce lze z pohledu programátora pohlížet na vyrovnávací stykové obvody jako na jistý druh paměti. Jinými slovy, spolupráce se stykovými obvody přídavných zařízení probíhá přes speciální paměť, tzv. *I/O paměť*. Tato I/O paměť je přístupná jak stykovým obvodům (jejichž součástí obvykle je), tak i ostatním prvkům mikropočítače přes společnou sběrnici. U mikroprocesorů, které mají speciální instrukce pro vstup a výstup (např. instrukce IN a OUT u mikroprocesorů firmy Intel), představuje I/O paměť speciální adresní prostor (obr. 2.20). Přepnutí mezi běžnou operační pamětí a I/O pamětí lze provést na základě indikace provádění instrukce pro vstup či výstup signálem IORQ.



Obr. 2.20. Speciální a mapovaná I/O paměť

Buňky I/O paměti se nazývají vstupní nebo výstupní registry a jsou identifikovány opět číslem, tzv. *I/O adresou*. Instrukce vstupu a výstupu pak představují vlastně speciální instrukce pro přesun, kde jeden z operandů je I/O adresa vstupního nebo výstupního registru.

Pro mikroprocesory, které nemají speciální instrukce vstupu a výstupu (např. mikroprocesory firmy Motorola), musí být I/O paměť vyčleněna

z běžného adresního prostoru. Jinými slovy, množiná I/O adres tvoří podmnožinu adresního prostoru operační paměti. Obvykle pak říkáme, že I/O paměť je *mapována do adresního prostoru* a mluvíme o *mapovaném vstupu* a *výstupu*. V dalším textu budeme používat označení vstupní, resp. výstupní registr pro buňky I/O paměti, bez ohledu na to, zda se nacházejí ve speciálním adresním prostoru, nebo jsou mapovány do základního adresního prostoru.

Stykový obvod přídavného zařízení si tedy lze z hlediska přístupu po sběrnici představit vždy jako sadu několika paměťových buněk – I/O registrů (často se pro I/O registry používá termín I/O port). Technickým připojením stykového obvodu je určeno umístění I/O registrů v rámci adresního prostoru I/O paměti – přidělení I/O adres. Principiálně má stykový obvod tři typy registrů:

– *řídicí registry*, do nichž lze zapisovat povely pro stykový obvod,

– *stavové registry*, z nichž lze číst informace o stavu stykového obvodu či přídavného zařízení a

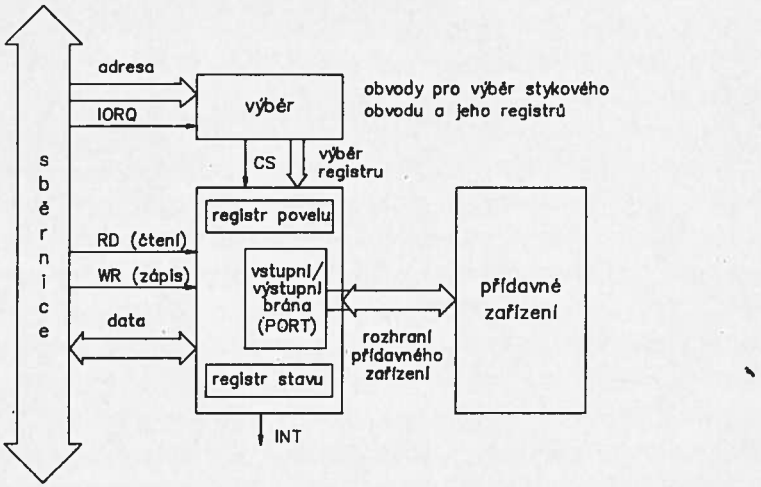
– *datové registry*, které slouží pro vlastní přenos dat.

Jednotlivých registrů může být ve stykovém obvodu více, podle jeho funkčních schopností. Stykový obvod často umí spolupracovat s několika podobnými typy zařízení. Část řídicích registrů proto obvykle slouží pro stanovení charakteristik přídavného zařízení a jejich nastavení se může provést např. při zahájení činnosti mikropočítače. Mluvíme pak o tzv. *programování stykových obvodů* a příslušné obvody se schopností nastavení označujeme termínem *univerzální programovatelné stykové obvody*.

Mimo registry I/O paměti zahrnují stykové obvody soubor technických prostředků pro přenos informace mezi přídavným zařízením a vlastními stykovými obvody (obr. 2.21). Z hlediska přídavného zařízení tvoří tyto prostředky bránu, přes kterou probíhá komunikace mezi zařízením a stykovými obvody. Často se rovněž používá termínu *vnější rozhraní*. Brána samozřejmě může sloužit buď pouze jako vstupní (pro přenos informace ze zařízení do stykových obvodů), nebo výstupní (pro přenos z obvodů na zařízení), příp. může být obousměrná. Charakteristiky bran lze často u programovatelných obvodů nastavit dynamicky zápisem do některého z řídicích registrů.

V jednoduchém případě představují stykové obvody pouze vyrovnávací paměť mezi sběrnicí a zařízením. To znamená, že brány tvoří přímo registry stykového obvodu (vnějším rozhraním je přímo I/O port). Ve

složitějších případech řídí činnost stykových obvodů vnitřní řadič na základě obsahu řídicích registrů, informací z řídicí sběrnice a vstupních bran. Složitost řadiče může být rozmanitá, od velmi jednoduchých až po řadiče, jejichž složitost je srovnatelná se složitostí samotného mikroprocesoru. Speciální stykové obvody jsou často schopny autonomně vykonávat náročné operace, související např. s ovládáním telekomunikačních linek, grafických zobrazovacích jednotek či diskových pamětí. V některých případech lze činnost řadiče stykového obvodu předepsat speciálním programem (tzv. kanálový program), který se nahraje do vnitřní paměti stykového obvodu. Mluvíme pak o tzv. *periferních procesorech*.



Obr. 2.21. Struktura stykového obvodu

Složitější stykové obvody jsou vyráběny jako samostatné integrované obvody, a to buď jako specializované, nebo univerzální. Univerzální programovatelné stykové obvody tvoří nepominutelnou součást mikropočítačových stavebnic. Ve struktuře stykového obvodu se často využívá vnitřní sběrnice pro komunikaci mezi řadičem, I/O pamětí a branami. Celý stykový obvod pak připomíná specializovaný mikropočítač. Někdy mohou být jednoduché stykové obvody zabudovány přímo do mikroprocesoru, jak je tomu v případě monolitických mikropočítačů (tzv. jednočipových).

2.6.2. Spolupráce se stykovými obvody

Spolupráce se stykovými obvody na straně mikropočítače probíhá obvykle ve čtyřech fázích. *První fáze*, kterou lze označit jako přípravnou, souvisí s nastavením programovatelných charakteristik stykových obvodů. U pevně nastavených obvodů tato fáze samozřejmě odpadá. Pro celou řadu zařízení, jejichž charakteristiky se během práce systému nemění, lze přípravnou fázi provést pouze jednou – při inicializaci systému. Ve *druhé fázi* je stykovým obvodům předán povel k požadované akci. *Třetí fáze* spočívá v kontrole činnosti stykových obvodů, závěrečná *čtvrtá fáze* slouží pro zhodnocení výsledků akce. Během třetí fáze může pracovat stykový obvod autonomně. Kontrola činnosti probíhá buď průběžně – čtením obsahu stavových registrů v programové čekací smyčce – nebo se očekává, že stykové obvody ohlásí konec třetí fáze (ať již úspěšný, či nikoliv) samy, např. žádostí o přerušení. Následná čtvrtá fáze dokončí akci, včetně případného přenosu výsledků mezi stykovými obvody a žadatelem na straně mikropočítače.

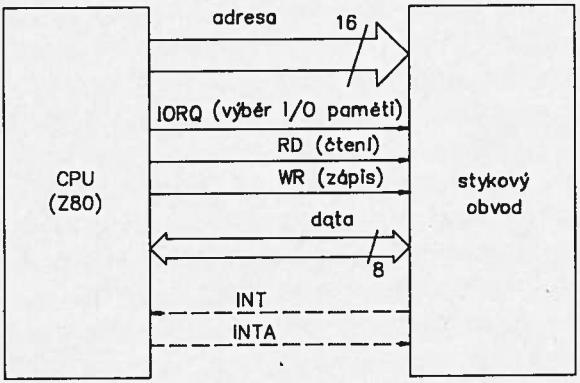
Komunikace po sběrnici systému probíhá podle standardních pravidel. Iniciátor akce si vyžádá přidělení sběrnice od řadiče sběrnice a postupně provede selekci zvolených I/O buněk vybraného stykového obvodu – přes adresní a příp. řídicí sběrnici. Po datové sběrnici vysílá patřičné obsahy I/O buňek – v první fázi charakteristiky, ve druhé povely pro stykový obvod včetně potřebných dat. Ve třetí fázi lze po datové sběrnici přenášet obsah stavových registrů, ve čtvrté pak případná získaná data.

Uvažme konkrétně případ, kdy chceme zapsat hodnotu uloženou v registru A mikroprocesoru do I/O buňky s adresou ADR. Instrukční repertoár mikroprocesoru může zahrnovat speciální instrukce pro komunikaci s I/O pamětí. Požadovaného efektu pak dosáhneme provedením instrukce typu OUT (ADR),A (mikroprocesor Z80), jejíž význam spočívá v přesunu obsahu střadače A na I/O adresu ADR. Buď přímo mikroprocesor, nebo doplňující obvody generují během provádění signál IORQ – indikaci, že se jedná o požadavek na přístup k I/O paměti. Pokud mikroprocesor nemá speciální instrukci OUT, provede se přesun běžnou instrukcí přesunu, např. MOVE A,ADR (mikroprocesor Motorola 68000).

V každém případě mikroprocesor požádá o přidělení sběrnice (pokud není řadičem sběrnice sám). Poté vyšle na adresní sběrnici adresu ADR, jejíž část (příp. v kombinaci se signálem IORQ) poslouží pro aktivaci

příslušného stykového obvodu a zbytek pro selekci I/O buňky v rámci stykového obvodu (dáno fyzickým zapojením výběrových obvodů). Obsah registru A je přenesen do datového registru mikroprocesoru (a tím i na datovou sběrnici) a přes datovou sběrnici zapsán do vybrané I/O buňky. Směr a okamžik přenosu zadává mikroprocesor po řídicí sběrnici, v tomto případě pomocí signálu WR.

K selekci stykového obvodu poznamenejme, že možným řešením je explicitní výběr skupinou samostatných výběrových vodičů, nebo lépe přes adresní sběrnici systému (nejsou nutné další vodiče). Výběrové obvody mohou být zapojeny pevně, příp. lze použít přepínačů (např. na desce modulu), kterými stanovíme umístění do adresního prostoru.

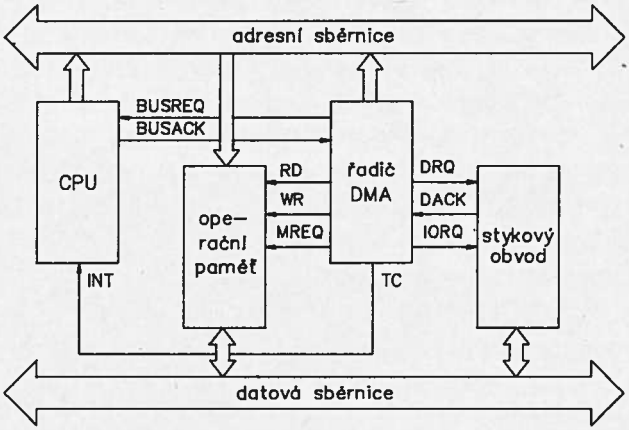


Obr. 2.22. Spolupráce mikroprocesoru se stykovými obvody

Z hlediska programového ovládání přídavných zařízení je výše uvedená činnost na sběrnici skryta – realizují ji technické prostředky. Přesto není na závadu vědět, jak komunikace se stykovými obvody principiálně probíhá. Programovým ovládáním přídavných zařízení přes stykové obvody se budeme zabývat podrobněji v následujících odstavcích. Prozatím postačí základní představa spolupráce se stykovými obvody (obr. 2.22). Mikroprocesor nejprve stykové obvody nastaví – naprogramuje – zápisem do některých řídicích registrů. Dále zapíše do (obecně jiných) řídicích registrů přes datovou sběrnici požadovaný povel. Poté již pracují stykové obvody autonomně a komunikují dohodnutým způsobem s fyzickým zařízením.

Mikroprocesor může číst obsah stavových registrů aby zjistil, zda periferní operace skončila, a v kladném případě přenést informaci po datové sběrnici. Jinou možností je ponechat pracovat stykové obvody samostatně a věnovat se jiné činnosti. Ukončení periferní operace je hlášeno speciálním signálem (např. READY), který je obvykle chápán jako žádost o přerušení a přiveden na vstup přerušovacího systému. Aktivace signálu READY pak způsobí přerušení činnosti mikroprocesoru a spuštění obslužného programu, který přenos dat uskuteční.

U velmi rychlých přídavných zařízení (např. disk) je, zejména pro delší přenosy, výhodnější pověřit přenosem řadič přímého přístupu do paměti (obr. 2.23). Data se pak přenášejí do paměti bez zásahu mikroprocesoru. Spolupráce stykového obvodu s řadičem pro přímý přístup probíhá obdobně jako s mikroprocesorem.



Obr. 2.23. Spolupráce řadiče DMA se stykovými obvody

2.6.3 Ovládání přídavných zařízení

V tomto odstavci probereme základní principy programového ovládání přídavných zařízení. Konkrétní příklady budou uvedeny v odstavci o přídavných zařízeních mikropočítačů (odst. 2.8).

Správa přídavných zařízení se skládá ze sady programů, které ovládají přídavná zařízení přes stykové obvody. Říkáme jim proto ovladače

přídavných zařízení (*drivers*). Protože musí být součástí každého programu, bývají zahrnuty do nejnižší vrstvy operačního systému. Ovladač musí umět spolupracovat jak se stykovými obvody, tak i znát rozhraní na straně zařízení. Z principu struktury stykových obvodů plyne, že ovladač bude obsahovat tyto části:

1. *inicializační část*, která slouží pro nastavení stykových obvodů a vlastní inicializaci ovladače,

2. *akční část* zajišťující zahájení a ukončení přenosu mezi počítačem a přídavným zařízením; akční část může aktivně čekat na ukončení přenosu nebo využívat přerušení – v tom případě zahrnuje ovladač ještě

3. *obsluhu přerušení*, provádějící vlastní přenos na základě aktivace přerušením.

Ovladač může být jednoúčelový, nebo využitelný pro určitou třídu zařízení. Ovladač obvykle zajišťuje přenos na úrovni informačních jednotek zařízení, tzn. že u znakových zařízení (klávesnice, tiskárna) přenáší informaci po znacích, u blokových zařízení (disk, páska) po blocích. Případné sjednocení způsobu ovládání různých zařízení nezajišťuje ovladač, ale vyšší vrstvy jádra operačního systému (správa virtuálních zařízení). Celkově lze předpokládat zhruba následující služby ovladače:

inicializace,
čtení informační jednotky zařízení,
zápis informační jednotky zařízení,
čtení stavu přenosu,
čtení parametrů ovladače,
nastavení parametrů ovladače.

V principu lze konstruovat dva typy ovladačů:

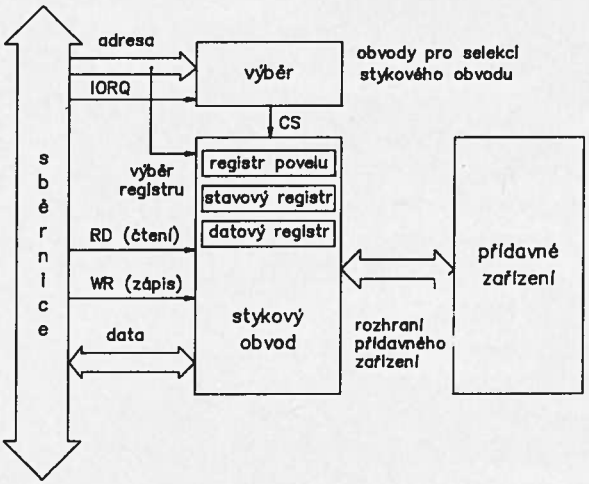
ovladač s aktivním čekáním nebo
ovladač využívající přerušení.

Je-li informační jednotka zařízení širší než adresní sběrnice systému, lze v ovladači využít obvody pro přímý přístup do paměti (zejména u rychlých zařízení) nebo zpracovávat informaci po informačních jednotkách.

2.6.3.1 *Ovladač s aktivním čekáním*

Na obr. 2.24 je naznačen princip technického zapojení pro *ovladač s aktivním čekáním*. Na straně přídavného zařízení jsou stykové obvody

přizpůsobeny konkrétnímu rozhraní zařízení. Na straně mikropočítače máme k dispozici běžnou sběrnici. Stykový obvod pro komunikaci s přídavným zařízením bude principiálně obsahovat registry:
- *datový registr*, který obsahuje přenášenou informaci,
- *řídicí registr* pro zápis povelu pro přenos,
- *stavový registr*, kde lze zjistit stav zpracování povelu.



Obr. 2.24. Princip technického zapojení pro ovladač s aktivním čekáním

Pokud využíváme univerzálních programovatelných stykových obvodů, je nutno mít na zřeteli fakt, že součástí takových obvodů bývá další *řídicí registr* nebo registry, jejichž obsah určuje aktuální parametry stykového obvodu. Jejich nastavení musí být provedeno před vlastním přenosem. Princip spolupráce se stykovým obvodem vyjadřuje následující program:

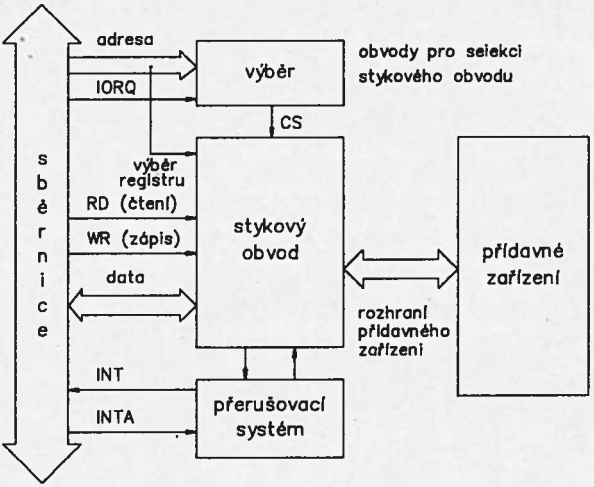
```
{služba: inicializace ovladače}
procedure INIT;
    begin
        "nastav parametry stykového obvodu"
    end;
```

```
{služba: výstup znaku}
procedure OUTCHAR (C: char);
    begin
        repeat
            "čti stavový registr"
            if "výstup je možný" then
                begin
                    "zapiš znak C do datového registru"
                    "vydej povel pro výstup"
                end
        until "výstup je možný"
    end;

{služba: zjištění stavu vstupu}
function INPSTAT: Boolean;
    begin
        "čti stavový registr"
        if "vstup připraven" then INPSTAT : = true
        else INPSTAT : = false;
    end;

{služba: vstup znaku}
procedure INPCHAR (var C: char);
    begin
        "vydej povel pro vstup"
        repeat until INPSTAT;
        "přenes obsah datového registru do C "
    end;
```

2.6.3.2 *Ovladač využívající přerušení*

Řešení uvedené v odst. 2.6.3.1 využívá programové čekací smyčky. Výhodnější je ovšem využít pro signalizaci připravenosti zařízení *přerušení*. Technické zapojení se využitím přerušení mírně zkomplikuje, neboť je nutno stykovým obvodem generovat žádost o přerušení a tu zpracovat

přerušovacím systémem. Situaci ilustruje obr. 2.25. Přerušovací systém bude zajišťovat předání žádosti stykového obvodu o přerušení na přerušovací vstup procesoru, včetně případné identifikace zdroje přerušení. Stykové obvody musí dále umožnit maskování této žádosti na základě požadavku procesoru, který nechce být rušen (např. během zpracování předchozího přerušení).



Obr. 2.25. Princip technického zapojení pro ovladač s přerušením

Abychom současně demonstrovali další techniky vytváření ovladačů, koncipujeme ovladač s přerušením pro přenos celého řetězu znaků. Z hlediska rychlosti přenosu je totiž výhodné, můžeme-li přenos dalšího znaku zahájit bezprostředně po skončení předchozího přenosu. Služba pro přenos bude mít jako parametry adresu pole znaků a jeho délku. Při návratu bude signalizovat případné chyby pomocí globální proměnné *RESULT*.

Oproti ovladači s aktivním čekáním přibývá v ovladači obsluha přerušení. Při inicializaci ovladače je navíc nutno nastavit obsluhu přerušení např. nahráním vhodných informací do příslušné položky přerušovacího vektoru (adresy počátku obslužného podprogramu).

Všechny části ovladače používají společné proměnné *ADRESA* a *DELKA*, které se na začátku nastaví dle parametrů služby a po přenosu

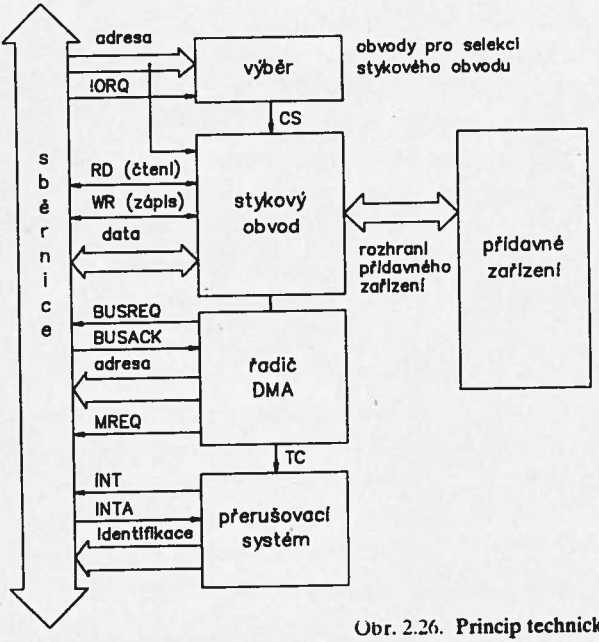
každého znaku aktualizují. Další změna proti aktivnímu čekání spočívá v zabránění opětovného vyvolání služby, neboť to je nyní principiálně možné – procesor nemusí být přidělen ovladači trvale po dobu zpracování.

```
{společné proměnné ovladače}
var ADRESA: ↑char ; {ukazatel na aktuální znak řetězu}
    DELKA: integer ; {délka řetězu}
    CINNY: Boolean ; {příznak aktivace ovladače}

{služba: inicializace ovladače}
procedure INIT;
    begin
        "zakaž přerušení procesoru"
        "nastav parametry stykového obvodu"
        "zakaž přerušení od stykového obvodu"
        "nastav přerušovací vektor na podprogram INTERRUPT"
        CINNY : = false;
        "povol přerušení procesoru"
    end;

{služba: výstup řetězu znaků}
procedure OUTSTR (A: ↑char; D: integer);
    begin
        "zakaž přerušení procesoru"
        if CINNY then RESULT : = 255
        else begin
            RESULT : = 0;
            CINNY : = true;
            "povol přerušení procesoru"
            ADRESA : = A {nastav parámetry pro přenos}
            DELKA : = D;
            "zakaž přerušení procesoru"
            "povol přerušení od stykového obvodu"
            "povol přerušení procesoru"
            while CINNY
            do; {čekej na dokončení přenosu}
```

```
            end;
            "povol přerušení procesoru"
        end;

{podprogram pro obsluhu přerušení}
procedure INTERRUPT;
{spustí se asynchronně na základě přerušení od přerušovacího
systému, které signalizuje připravenost zařízení pro přenos;
v okamžiku spuštění je další přerušení maskováno}
        begin
            "uschovej stav procesoru"
            "zakaž přerušení od stykového obvodu"
            if not CINNY then RESULT : = 254
            else begin
                RESULT : = 0;
                if (DELKA = 0) then CINNY : = false
                else begin
                    "povol přerušení procesoru"
                    "vlož znak ADRESA↑ do datového
                    registru stykového obvodu"
                    "vydej povel pro přenos"
                    DELKA : = DELKA - 1;
                    ADRESA : = ADRESA + 1;
                    "zakaž přerušení procesoru"
                    "povol přerušení od stykového obvodu"
                end
            end;
            "obnov stav procesoru"
            "povol přerušení procesoru"
            "návrat do přerušeného procesu"
        end;
```

Vlastní přenos znaků obstarává procedura *INTERRUPT*, dle nastaveného obsahu proměnných *ADRESA* a *DELKA*. Spuštění procedury *INTERRUPT* zajistí služba *OUTSTR* povolením přerušení od stykového obvodu. Přerušením hlásí stykový obvod připravenost přídavného zařízení.

Přerušení od stykového obvodu může nastat bezprostředně po jeho povolení, a proto v této kritické sekci zakážeme možnost přerušení procesoru. Tím zajistíme, aby se akce povolení přerušení od stykového obvodu vždy správně dokončila. Stejným způsobem zajistíme korektní přístup programových jednotek ke společné proměnné *CINNY* tak, aby mezi testem obsahu této proměnné a jejím případným nastavením nemohlo dojít k nežádoucí interakci.



Obr. 2.26. Princip technického zapojení pro ovladač s přímým přístupem

Podobně při aktivaci obsluhy přerušení zakážeme další přerušení od stykového obvodu, dokud nedokončíme obsluhu přerušení předchozího. Podprogram pro obsluhu přerušení může být aktivován během provádění jiných procesů, proto pozorně uschováme stav procesoru pro správný návrat do stavu při příchodu žádosti o přerušení. Obnovu stavu procesoru před návratem zahrneme do kritické sekce, aby zbytečně nedocházelo k přeplnění zásobníku.

Po nastavení parametrů a vydání povelu pro přenos čeká procedura *OUTSTR* ve smyčce na dokončení služby. V multiprogramovém systému by

zde bylo možné předat řízení zpět volajícímu procesu. To je principiální přínos přerušení, který se však v monoprogramových verzích systému CP/M nevyužívá.

2.6.3.3 *Ovladač s přímým přístupem*

Pro velmi rychlá přídavná zařízení jako např. disk je výhodnější koncipovat stykové obvody s využitím přímého přístupu do paměti. Do stykových obvodů proto zařadíme řadič pro přímý přístup do paměti a vlastní přenos informace zajistíme nastavením (naprogramováním) parametrů řadiče. Ukončení přenosu lze indikovat přerušením nebo snímat stav přenosu v programové čekací smyčce. Princip technického zapojení udává obr. 2.26, kde přibývají signály BUSREQ a BUSACK pro spolupráci řadiče přímého přístupu s procesorem při sdílení sběrnice. Princip ovladače podává následující program.

```
{společné proměnné}
var CINNY: Boolean ; {příznak aktivace přenosu}

{služba: inicializace ovladače}
procedure INIT;
      begin
            "zakaž přerušení procesoru"
            "nastav parametry stykového obvodu"
            "zakaž přerušení od stykového obvodu"
            "nastav přerušovací vektor na podprogram INTERRUPT"
            CINNY : = false;
            "povol přerušení procesoru"
      end;

{služba: přenos řetězu znaků}
procedure OUTSTR (A: ↑char; D: integer);
      begin
            "zakaž přerušení procesoru"
            if CINNY then RESULT : = 255
```

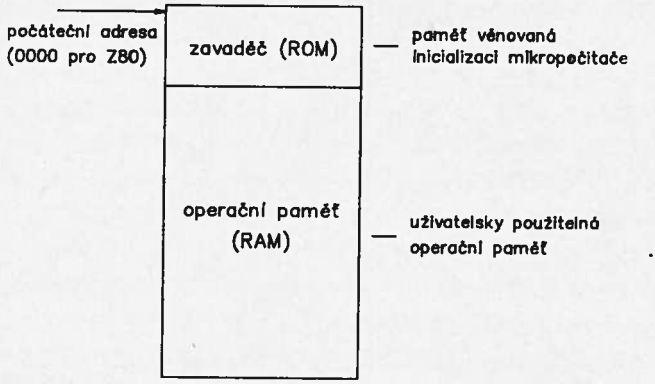
```
            else begin
                    RESULT : = 0;
                    CINNY : = true ;
                    "nastav parametry řadiče pro přímý přístup
                    do paměti dle hodnot A, D a vydej povel pro
                    zahájení přenosu"
                    "povol přerušení od stykového obvodu"
                    "povol přerušení procesoru"
                    while CINNY do; {čekej na dokončení přenosu}
                end;
                "povol přerušení procesoru"
        end;

{podprogram pro obsluhu přerušení}
procedure INTERRUPT;
{spustí se asynchronně na základě přerušení od přerušovacího
systému, které signalizuje dokončení přenosu řadičem DMA;
v okamžiku spuštění je další přerušení maskováno}
        begin
                "uschovej stav procesoru"
                "zakaž přerušení od stykového obvodu"
                if not CINNY then RESULT : = 254
                else begin
                        RESULT : = 0;
                        CINNY : = false ;
                end;
                "obnov stav procesoru"
                "povol přerušení procesoru"
                "návrat do přerušeného procesu"
        end;
```

2.7 Startování mikropočítače

Po připojení mikropočítače k napájecímu zdroji zajistí zpravidla technické prostředky nastavení všech složek (obvodů) do určitého *výchozího stavu*. Při použití obvodů vyšší integrace lze počátečního nastavení

dosáhnout *generováním signálu inicializace* (RESET) pro všechny použité obvody na základě impulsu z napájecího zdroje. Některé mikropočítače jsou vybaveny startovacím tlačítkem (INIT, RESET apod.), kterým rovněž uvedeme systém do správného počátečního stavu. Signál ze startovacího tlačítka je opět vyveden na inicializační vstupy obvodů. Jedním z nich je i vlastní mikroprocesor, který se tím uvede do definovaného stavu – zejména se do čítače adres zavede počáteční adresa, z níž si mikroprocesor začne vybírat instrukce k provádění. Konstruktér mikropočítače musí zařídit, aby v tuto chvíli byla na počáteční adrese posloupnost instrukcí, která zahájí činnost mikropočítače a uvede celý systém do užitného stavu.

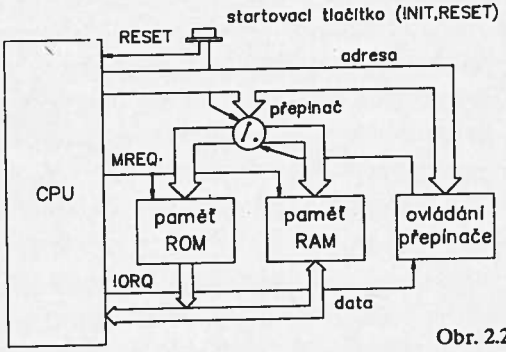
Inicializační sekvence instrukcí musí být uložena v operační paměti. Protože paměť typu RWM nemá definován obsah při připojení napájení, bývá pro tyto účely použita permanentní paměť (typu ROM). V ní je nahrán inicializační program, který se má spustit při startování činnosti mikropočítače. Aby se skutečně spustil, musí být paměť ROM zapojena tak, že začátek inicializačního programu souhlasí s počáteční adresou mikroprocesoru (obr. 2.27).



Obr. 2.27. Umístění inicializačního programu

Úkolem *inicializačního programu* bývá počáteční nastavení všech složek mikropočítače, zejména nastavení programovatelných stykových obvodů (aby mikropočítač mohl komunikovat s přídavnými zařízeními), příp.

nastavení některých buněk operační paměti (zejména přerušovacího vektoru, aby zpracování přerušení bylo deterministické) apod. Často je součástí inicializačního programu řada programových testů (testy operační paměti, stykových obvodů, či vnějších zařízení). U většiny systémů spočívá další funkce inicializačního programu v zavedení operačního systému z vnější paměti (např. disku) do paměti operační. Zavádění operačního



Obr. 2.28. Příklad realizace stínové paměti

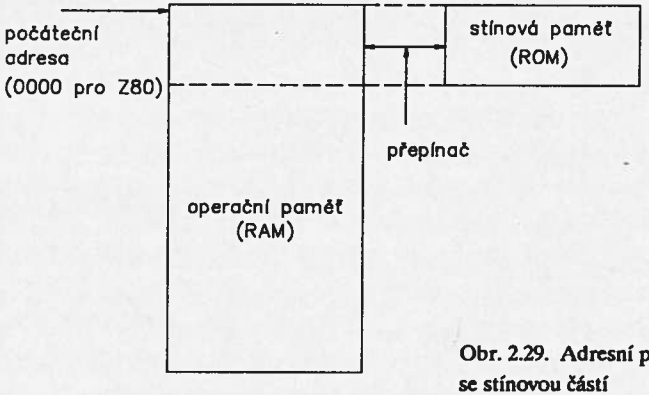
systému provádí tzv. *zavaděč* (bootstrap), který pak předá další řízení operačnímu systému.

Umístění operačního systému (nebo jeho podstatné části) na vnější médium je nutné zejména při větším rozsahu programů operačního systému (kvůli úspoře operační paměti) a v neposlední řadě z důvodů snadné modifikovatelnosti (umístíme-li celý operační systém do pevných pamětí, souvisí každá jeho modifikace s technickým zásahem do mikropočítače). Naproti tomu umístění operačního systému na vnější médium způsobuje jisté zpomalení přístupu k programům. Volba vnějšího média je proto velmi důležitá a jeho charakteristiky silně ovlivňují schopnosti operačního systému. Označení hostitelského média pak přenášíme i do označení operačního systému, např. diskový operační systém hostuje na diskovém médiu.

Inicializační program je uváděn v činnost pouze příležitostně (při uvádění mikropočítače do činnosti, při obnovení činnosti po neopravitelné chybě atd.). Příslušnou část adresního prostoru paměti však nelze běžně používat (dá se z ní jen číst). U systémů, kde je omezení rozsahu použitelné paměti nežádoucí (zejména tedy u 8bitových systémů), bývá proto použito triku, pomocí něhož lze uživatelsky využívat celý adresní prostor. Na stejné

adresy jako paměť ROM je přes přepínač připojena i paměť RWM (obr. 2.28). Přepínač obou pamětí je řízen jednak startovacím tlačítkem (INIT) a jednak speciálním stykovým obvodem jako zvláštní přídavné zařízení.

Při stisku tlačítka INIT je přepínač přepnut na paměť typu ROM, z níž pak mikroprocesor vybírá instrukce inicializačního programu a provádí je. Poslední instrukcí inicializačního programu je požadavek na ovladač pře-



Obr. 2.29. Adresní prostor se stínovou částí

pínače, který přepne přepínač a připojí na adresní sběrnici paměť typu RWM. Paměť ROM je od tohoto okamžiku zamaskována až do dalšího stisku tlačítka INIT. Často se jí proto říká *stínová paměť*. Adresní prostor se stínovou částí je zobrazen na obr. 2.29.

2.8 Přídavná zařízení mikropočítačů

Mozkem mikropočítačového systému je mikropočítač – procesor, vnitřní paměť a stykové obvody přídavných zařízení. Jakákoliv aktivita mikropočítače se však může navenek projevit pouze pomocí vnějších přídavných zařízení. Odtud vstupují data a většinou i programy, na přídavných zařízeních se zobrazují výsledky práce mikropočítače. Přídavná zařízení proto svými schopnostmi limitují vnější projevy mikropočítače. V dalším textu stručně probereme typická přídavná zařízení mikropočítačů. Pro lepší orientaci je rozdělíme na zařízení určená pouze pro vstup informace,

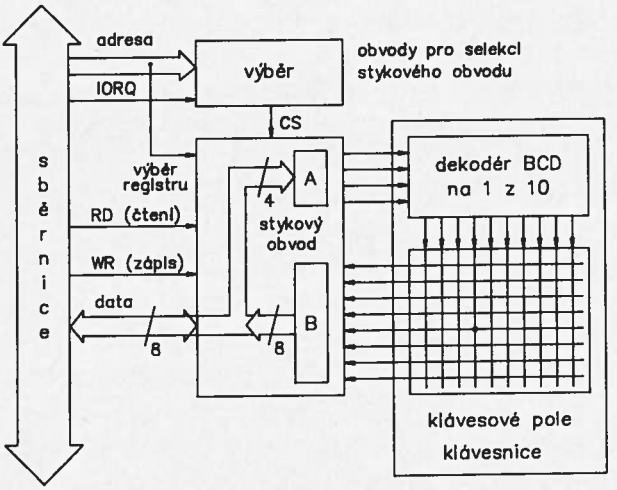
zařízení sloužící pouze pro výstup informace a zařízení dovolující jak vstup, tak i výstup informace. Ta ještě rozlišíme na vnější paměti a zařízení určená pro komunikaci.

2.8.1 Vstupní přídavná zařízení

2.8.1.1 *Klávesnice*

Klasickým vstupním zařízením mikropočítače je klávesnice. Klávesnice slouží pro *vstup textové nebo numerické informace* a může být využívána i pro *vstup různých řídicích povelů*. Z hlediska programového ovládání představuje klávesnice vstupní zařízení, které je schopno dodat informaci ve formě *posloupnosti číselných kódů* z jistého rozsahu.

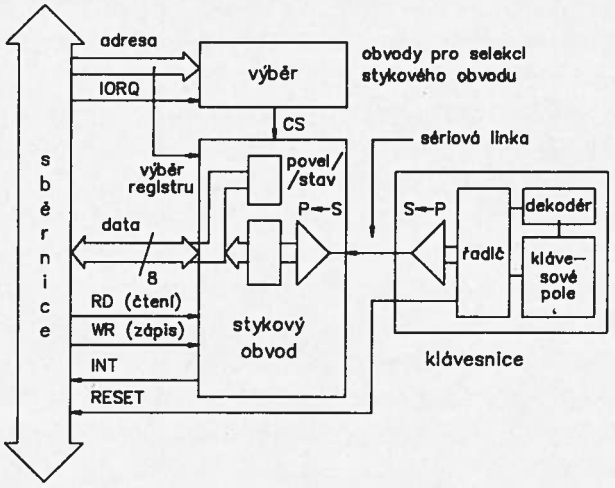
Vlastní technické řešení klávesnice obvykle spočívá v uspořádání kláves do matice tak, že stiskem klávesy způsobíme propojení odpovídajícího sloupce a řádku v matici. Postupným přivedením signálu např. na jednotlivé sloupce matice dostáváme v okamžiku výběru správného sloupce rovněž indikaci řádku. Zpracováním těchto informací získáme číselný kód klávesy.



Obr. 2.30. Stykové obvody klávesnice počítače MZ–800

V jednoduchém případě je klávesnice připojena jako pasívní zařízení, které na základě dodaného čísla sloupce vrátí informaci o řádcích, ve kterých byla v daném sloupci stisknuta klávesa (obr. 2.30). Postupný výběr sloupců a kódování kláves jsou zajištěny programově.

Moderní klávesnice bývají vybaveny vlastním procesorem, který jednak řídí snímaní a kódování kláves, a současně zajišťuje komunikaci po vnějším rozhraní se stykovými obvody. Vnější rozhraní pak může být tvořeno sériovou linkou, kterou se budeme zabývat v odstavci o komunikačních prostředcích. Stykové obvody pro připojení klávesnice pak obvykle vytvářejí programový model, obsahující jednobitový řídicí a stavový registr a vícebitový datový registr (obr. 2.31). Do řídicího registru zapisujeme povel "snímej", který způsobí, že při stisku jakékoliv klávesy je nastaven ve stavovém registru příznak "sejmuto" a do datového registru zapsán číselný kód stisknuté klávesy. Stykové obvody mohou současně generovat žádost o přerušení, kterou lze využít pro přenos kódu do operační paměti. Jinou možností je programové testování stavového příznaku. Následující povel "snímej" způsobí shození příznaku "sejmuto" a opakování činnosti.



Obr. 2.31. Stykové obvody klávesnice počítače IBM PC

U nejlevnějších mikropočítačových stavebnic či ve specializovaných systémech bývá použita hexadecimální klávesnice obsahující tlačítka pro hexadecimální číslice a několik funkčních tlačítek pro zadávání povelů. Hexadecimální klávesnice bývá využívána pro přímé zadávání adres a obsahů buněk operační paměti. Funkční tlačítka slouží např. pro rozlišení zadání adresy a obsahu, přechodu mezi režimem nastavení obsahu paměti a spouštěním takto vytvořeného programu apod.

Pro domácí počítače je typické použití malé klávesnice s plnou sadou základních abecedněčíslicových znaků. Pomocí přeřazovačů lze dosáhnout zdánlivého zvýšení počtu kláves tak, že každé tlačítko může mít několik významů. Obvykle však počet všech kombinací nepřesáhne 256 a pro všechny významy postačí k reprezentaci 256 kombinací kódů. Klávesnice pak dodává do mikropočítače jednu slabiku při stisku smysluplné kombinace kláves.

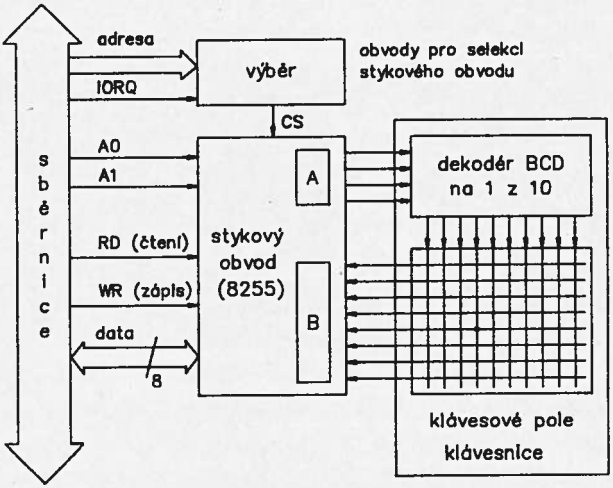
Podobná je situace u jednodušších osobních počítačů, kde se rovněž využívá klávesnice s plnou sadou abecedněčíslicových znaků a přeřazovači. Na rozdíl od domácích počítačů je provedení klávesnice kvalitnější a repertoár přeřazovačů menší. Odstraňuje se tím poněkud "šamanský" způsob manipulace s klávesnicí domácího počítače. Nejčastěji pak dodává klávesnice 128 kombinací standardního znakového kódu ASCII (American Standard Code for Information Interchange, viz Příloha 7.1).

Typická klávesnice současného osobního počítače obsahuje jednočipový mikropočítač, který ji řídí. Klávesnice dodává pouze informaci o tom, která klávesa byla právě stisknuta či uvolněna. Každá klávesa má přiřazeno jedno ze 128 identifikačních čísel, ke kterému je doplněna jednobitová informace, zda byla klávesa stisknuta či uvolněna. Stiskneme-li např. klávesu číslo 30 (označenou symbolem 'A'), předává klávesnice kód 1EH, po uvolnění pak kód 9EH. Programově je nutno obsloužit, zda se má jednat o znak 'A', 'a' či CTRL-A, v závislosti na předchozím stisku přeřazovačů. Klávesnice navíc nastavuje, při současném stisku kláves CTRL, ALT a DEL, speciální signál RESET, kterého lze využít (obdobně jako samostatného tlačítka INIT) pro havarijní znovunastartování počítače.

Klávesové pole klávesnice bývá rozděleno na tři části: abecedněčíslicovou, číslicovou a funkční. *Abecedněčíslicová* část představuje běžnou klávesnici, *číslicová* zahrnuje vstup cifer a současně tlačítka pro pohyb kurzoru a tlačítka pro volbu základních editačních funkcí. *Funkční* část klávesnice představuje sadu tlačítek, která jsou určena pro snadné vyvolávání

různých funkcí. Přiřazení funkcí tlačítkům je určeno konkrétním programem, přičemž lze funkční klávesnici opatřit výměnným předznačením dle používaného programu. Termín funkční klávesnice se používá rovněž pro označení možnosti přiřadit klávesám různý význam (např. text, který daná klávesa bude zastupovat). Některé osobní počítače mají funkční klávesnici uspořádanou tak, aby bylo možné aktuální význam funkčních kláves znázornit na obrazovce bezprostředně nad klávesami.

Klávesnice tvoří postačující vybavení pro vstup informace do mikropočítače. Pro zvýšení pohodlí při interakci s mikropočítačem lze tuto základní sestavu doplnit o další vstupní zařízení, která usnadňují zejména rychlou identifikaci místa na ploše obrazovky, případně instinktivní zadání směru pohybu po obrazovce.



Obr. 2.32. Technické řešení stykových obvodů klávesnice počítače MZ-800

2.8.1.2 *Programové ovládání klávesnice*

Uvažujme pro ilustraci příklad jednoduchého ovladače klávesnice pro počítač MZ-800. Klávesnice je připojena přes univerzální programovatelný stykový obvod 8255 podle obr. 2.32. Rozhraní klávesnice se skládá ze čtyř

vstupních vodičů, kterými se zadává jeden z deseti sloupců klávesového pole, dále z osmi výstupních datových vodičů, které jsou připojeny na řádky klávesového pole. Na straně mikropočítače máme k dispozici systémovou sběrnici.

Univerzální programovatelný stykový obvod 8255 je zapojen tak, že jsou využity dvě brány na straně rozhraní klávesnice. Brána A je výstupní 4bitová, brána B je vstupní 8bitová. U obvodu 8255 dosáhneme příslušné konfigurace bran zápisem povelu 130 = 10000010B = 82H do řídícího registru R obvodu 8255.

Na straně mikropočítače připojíme datový registr D (vyrovnávací paměť) na datovou sběrnici. Dále připojíme signály RD a WR řídicí sběrnice na odpovídající vstupy obvodu 8255, čímž umožníme řízení toku dat směrem do obvodu 8255 (WR) nebo z obvodu 8255 (RD). I/O paměť obvodu 8255 tvoří 4 buňky – řídicí registr R a brány A,B,C. Jejich selekce se provádí vstupními signály A0 a A1 obvodu 8255 dle tabulky:

A1	A0	Význam
0	0	brána A
0	1	brána B
1	0	brána C
1	1	řídicí registr R

Celková aktivace obvodu se provádí vstupním signálem CS (Chip Select). Předpokládejme, že používáme mikroprocesor Z80, který má oddělený I/O adresní prostor o celkovém možném rozsahu 256 I/O adres. Zbývá tedy přidělit registrům obvodu 8255 určité I/O adresy, např. dle tabulky:

I/O adresa	Registr 8255
0D0H	brána A
0D1H	brána B
0D2H	brána C
0D3H	řídicí registr R

Připojení na adresní sběrnici provedeme tak, že adresní vodiče A0 a A1 připojíme přímo na vstupy obvodu 8255. Vodiče A2 až A15 adresní

sběrnice přivedeme spolu se signálem IORQ na vstup výběrového obvodu, který v případě, že na adresní sběrnici je kombinace 00000000110100XX, tj. některá z adres 0D0H až 0D3H, a je nastaven signál IORQ (provádí se instrukce IN nebo OUT), aktivuje obvod 8255 signálem CS. Nyní již známe všechny potřebné údaje pro vytvoření ovladače klávesnice. Pro jednoduchost nebudeme v ovladači řešit problémy kolize při stisku více kláves současně, ale budeme předpokládat, že v daném okamžiku může být stisknuta pouze jedna klávesa. Podobně vypustíme obsluhu přeřazovačů. Akční část ovladače bude čekat na stisk klávesy; uvolnění klávesy ignoruje. Pro srovnání aktuálního stavu klávesnice a naposledy čteného stavu využívá ovladač pomocné pole KEYST, které obsahuje původní stav klávesnice.

```
; KOSTRA JEDNODUCHÉHO OVLADAČE KLÁVESNICE
; DEFINICE SYMBOLICKÝCH KONSTANT
PORTA   EQU 0D0H        ; I/O adresa brány A
PORTB   EQU 0D1H        ; I/0 adresa brány B
REGR    EQU 0D3H        ; I/0 adresa řídicího registru
INITC   EQU 082H        ; konfigurační povel pro 8255
; PAMĚŤ STAVU KLÁVESNICE
KEYST:  DS  10
; INICIALIZAČNí ČÁST (služba INIT)
INIT:LD  A,INITC        ; konfigurace bran obvodu 8255
     OUT (REGR),A
     LD  HL,KEYST       ; nulování stavu klávesnice
     LD  B,10
INI1: LD  (HL),0
     INC HL
     DJNZ INI1
     RET
; AKČNÍ ČÁST (služba čtení z klávesnice)
; PARAMETRY:
; vstup:    -
; výstup:   registr C - kód klávesy
RDKEY:  LD  HL,KEYST ; HL - ukazatel na předchozí stav
        LD  BC,10       ; počet sloupců —> B, 0 —> C
```

```
RD1:    LD   A,C          ; identifikace sloupce
        OUT (PORTA),A ; zadej sloupec přes bránu A
        NOP               ; pozdržení
        IN   A,(PORTB) ; přečti obsah brány B
        CP   (HL)         ; porovnej s předchozím stavem
        JR   Z,RD2        ; pokud nenastala změna —> RD2
        LD   (HL),A       ; jinak zapamatuj nový stav
        OR   A            ; otestuj uvolnění klávesy
        JR   NZ,RD3       ; došlo ke stisku klávesy —> RD3
RD2:    INC  HL           ; další slabika stavu
        INC  C            ; další sloupec
        DJNZ RD1
        JP   RDKEY        ; čekej na změnu na klávesnici
RD3:    LD   B,C          ; spočti kód klávesy
        LD   A,(HL)       ; řádek
RD4:    ADD A,8           ; přičti 8*sloupec
        DJNZ RD4
        SUB 8
        LD   C,A          ; předej výsledek v registru C
        RET
```

2.8.1.3 *Myš*

Jednoduché zadání směru pohybu po ploše obrazovky dovoluje zařízení, které umí snímat pohyb lidské ruky ve dvou navzájem kolmých směrech – osách. Nejčastější provedení pro mikropočítače má tvar krabičky, na jejíž spodní straně jsou otočné prvky pro snímání pohybu. Krabička je spojena se stykovými obvody kabelem – celkový vzhled jí předurčil označení *myš*.

Principiálně se jedná o *dva nezávislé inkrementální snímače pohybu*, které mohou být realizovány např. jako opticky řízené reverzibilní čítače, příp. analogově. Myš bývá vybavena nejméně jedním tlačítkem, kterým indikujeme začátek a konec pohybu (tzv. režim tažení myši – tlačítko zůstává stisknuto po dobu pohybu). Krátkým stiskem tlačítka provádíme potvrzení, identifikaci nebo výběr (tzv. režim signalizace). Snímání opakovaných krátkých stisků nahrazuje často nepřítomnost více tlačítek.

Po připojení přes stykové obvody, představuje myš tři nezávislé vstupní registry, jejichž obsah se mění pohybem myši a stiskem tlačítek. Pohybujeme-li např. myší doprava, mohou technické prostředky zajistit inkrementaci čítače v ose *x* pokaždé, urazí-li myš určitou vzdálenost. Podobně při pohybu vlevo, je obsah čítače v ose *x* snižován. Současně se snímá i pohyb v ose *y*. Stav tlačítek je udržován v samostatném registru, příp. je stisk tlačítka indikován žádostí o přerušení.

Při spolupráci s myší je programově nebo technicky snímán obsah vstupních registrů v určitých časových intervalech a získané informace lze využít například pro pohyb kurzoru po obrazovce. Tato vizuální zpětná vazba je velmi důležitá, neboť bez ní by bylo využití myši problematické.

Nejčastěji se myš používá u osobních počítačů pro rychlý pohyb v textu či obrázku a pro rychlý výběr z nabízeného "menu" možností. Lze ji však využít i pro umísťování obrazců na ploše obrazovky nebo kreslení čar.

2.8.1.4 *Světelné pero*

Světelné pero je vstupní zařízení vyvinuté pro účely počítačové grafiky. Proto je jeho použití u mikropočítačů většinou omezeno na pracovní stanice zaměřené na grafické aplikace, obvykle ve spojení s rastrovou zobrazovací jednotkou. Využívá se zde způsobu zobrazení řádkovaným elektronovým paprskem. Světelné pero je vybaveno fotocitlivým prvkem, který po přiblížení pera k obrazovce indikuje okamžik průchodu elektronového paprsku. Na základě této informace dokáže řadič zobrazovací jednotky určit bodový řádek a sloupec, ve kterém došlo k indikaci.

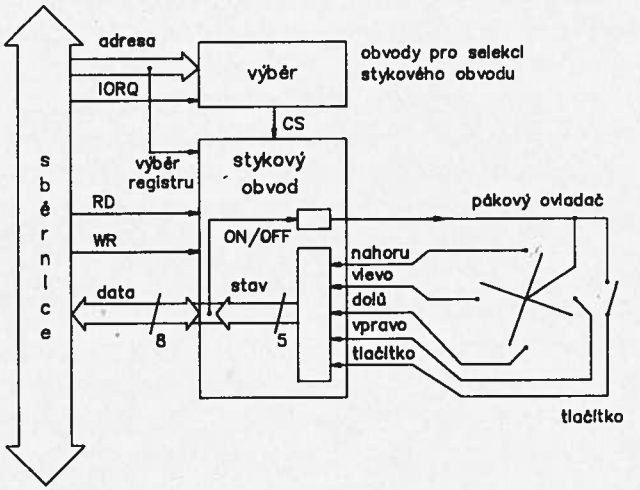
Světelné pero bývá rovněž vybaveno tlačítkem, kterým lze signalizovat například ukončení výběru apod. Na rozdíl od myši lze světelným perem přímo ukazovat na určité místo obrazovky – myší se musíme přesunout. Programový styk se světelným perem je tedy obdobný jako u myši, pouze údaje v registrech pro osu *x* a *y* vyjadřují aktuální absolutní polohu pera, nikoliv relativní posuv vzhledem k předchozí poloze.

2.8.1.5 *Pákový ovladač (joystick)*

Pákový ovladač má obdobnou funkci jako myš, ale vzhledem připomíná spíše řídicí páku pilota. Nejčastěji je řešen tak, že dovoluje indikaci

jednoho z osmi možných směrů a signalizaci stisku tlačítka. Bývá často používán u domácích počítačů pro ovládání různých her.

Možné řešení stykových obvodů pro připojení pákového ovladače je naznačeno na obr. 2.33. Programový model obsahuje 1bitový řídicí registr a 5bitový datový registr. Po zápisu povelu "snímej" do řídicího registru je možno číst obsah datového registru, kde se nastavují indikace ze spínačů.



Obr. 2.33. Stykové obvody pákového ovladače počítače MZ-800

2.8.1.6 *Dotyková obrazovka*

Snaha využít k interakci přímo informaci zobrazenou na obrazovce vedla k realizaci tzv. *dotykových obrazovek*, kde identifikaci polohy udáváme přímým ukázáním na obrazovku. Jako ukazovátko lze obvykle použít libovolný vhodný předmět, například i lidský prst. Indikace polohy se nejčastěji provádí optickým snímáním pomocí infračerveného rastru před stínítkem obrazovky, nebo akusticky, pomocí ultrasonických vysílačů a přijímačů. Určitou nevýhodou tohoto způsobu indikace je poměrně malá rozlišovací schopnost. Způsob programového ovládání je podobný jako u světelného pera.

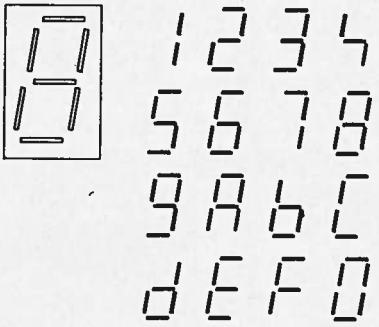
2.8.1.7 *Snímač souřadnic*

Vstupní grafické možnosti mikropočítače někdy doplňuje tzv. *snímač souřadnic* (tablet), obvykle v jednodušším provedení. V principu se jedná o zařízení, které různým způsobem snímá polohu ukazovátka v jistém souřadném systému. Využívá se např. induktivní snímání polohy cívky nad mřížkou vodičů, nebo mechanického spojení ukazovátka s dvěma kolmými lineárními snímači pohybu apod. Výstupy snímačů jsou upraveny do číslicového tvaru a přes stykové obvody přístupné programům. Podobně jako u myši, pákového ovladače či světelného pera je ukazovátko vybaveno signálním tlačítkem. Na rozdíl od myši se však nesnímá směr pohybu, ale poloha. Ovládání je proto podobné světelnému peru.

2.8.2 Výstupní přídavná zařízení

2.8.2.1 *Zobrazovací jednotky*

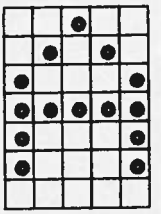
Nejjednodušší zobrazovací jednotkou mikropočítačů je tzv. *segmentový displej.* Jeho použití je typické pro nejprimitivnější třídu mikropočítačů, příp. se používá jako jednoduchá signalizační zobrazovací jednotka. Segmentový displej se skládá z několika zobrazovacích prvků (segmentů), které lze samostatně rozsvítit (příp. zvýraznit proti pozadí). Rozsvícením určité kombinace segmentů dosáhneme zobrazení jednoho z možných obrazců, podle prostorového rozmístění segmentů. Nejčastěji se používá sedmi segmentů



Obr. 2.34. Sedmisegmentový displej a jeho abeceda

uspořádaných podle obr. 2.34. Z možných 128 kombinací bývá využíváno pouze 16 kombinací, odpovídajících šestnáctkovým číslicím. Několik sedmisegmentových displejů pak dovoluje zobrazení několikamístného šestnáctkového čísla. Často proto bývá používán pro zobrazení obsahu vnitřní paměti mikropočítače, obvykle ve formě dvojice <adresa : obsah>.

Kvalitativně lepší zobrazení vyžaduje zvýšení počtu zobrazovacích prvků, např. je možno použít bodový rastr. Z bodů je pak možno sestavit postačující počet dostatečně rozlišitelných symbolů například pro zobrazení všech znaků jisté abecedy (obr. 2.35). Jako dostatečná byla původně zvolena



Obr. 2.35. Bodová matice znaku

v anglicky mluvících zemích matice 5×7 bodů, z nichž lze sestavit libovolný znak anglické abecedy. Pro slovansky píšící země by pravděpodobně lépe vyhovovala matice o něco větší. Smysluplným kombinacím bodů jsou přiřazeny odpovídající kódy znaků.

Pro zobrazení bodové matice se využívá stínítka obrazovky, které dává dostatek prostoru pro zobrazení více znaků. U domácích počítačů se často využívá jako zobrazovací jednotka běžný *televizní přijímač*. Výstupní obvody mikropočítače musí zajistit generování úplného televizního signálu, ve kterém se modulací jasu zajistí zobrazení požadované bodové matice. Ovládací obvody zobrazovací jednotky vybírají kódy zobrazovaných znaků z tzv. *obrazové paměti* (video RAM), transformují na bodové matice, linearizují do posloupnosti, kterou modulují jas základního televizního signálu. Opakovaným výběrem z obrazové paměti se dosáhne stabilizace obrazu na stínítku. Výstup na zobrazovací jednotku pak neznamená nic jiného, než zápis kódu znaku do obrazové paměti.

Použití televizního přijímače jako zobrazovací jednotky mikropočítače je výhodné pro snadnou dostupnost a možnost barevného zobrazení. Řada domácích počítačů (Sinclair, Atari) obsahuje speciální zákaznické stykové obvody pro ovládání zobrazovací jednotky, dovolující různá kouzla s obra-

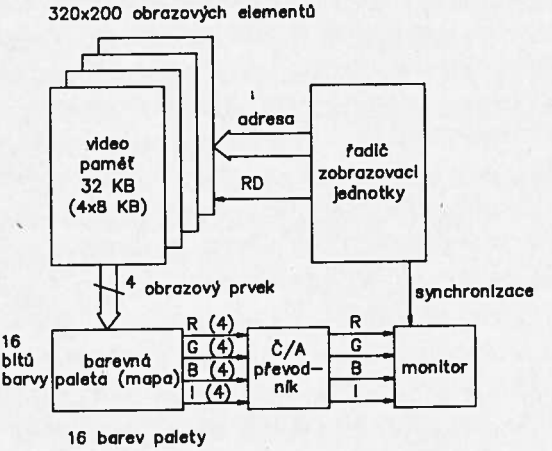
zem. Přesto má použití běžného televizního přijímače limitující faktory. Především nelze zvýšit rozlišovací schopnost danou parametry televizního signálu a parametry běžné obrazovky. Rovněž modulace a zpětná demodulace signálu v televizním přijímači může způsobit zhoršení kvality obrazu. Proto bývá u osobních počítačů použit jako zobrazovací jednotka speciální zobrazovací monitor, kde odpadá konverze na úplný televizní signál a zpět.

Zobrazovací monitor pracuje podobným způsobem jako zobrazovací obvody televizního přijímače. Elektronový paprsek probíhá postupně jednotlivé bodové řádky obrazovky. Jeho modulací měníme jas bodů na stínítku. Na rozdíl od běžného televizního přijímače však můžeme při kvalitní obrazovce a přesnějším vychylovacím systému dosáhnout vyšší rozlišovací schopnosti a stálosti obrazu. Např. běžný televizní monitor dovoluje zobrazení přibližně 250×400 bodů. To odpovídá zhruba rastru 16×64 znaků. Běžný zobrazovací monitor osobního počítače typu IBM PC umí zobrazit 348×720 bodů, což odpovídá rastru 25×80 znaků, při rozsahu matice znaku 14×9 bodů. Velmi kvalitní současné monitory, používané u pracovních stanic, zobrazují 1 024×1 280 bodů. Rastrový charakter zobrazení na stínítku obrazovky dovoluje vykreslení libovolné kombinace bodů. To dovoluje použít televizní zobrazovací jednotku nejen pro výstup abecedněčíslicové informace (textu), ale i pro grafický výstup. Často je možné oba typy zobrazení (znaky a bodový obrázek) kombinovat. Někdy lze využít dokonce několika zobrazovacích rovin současně, přičemž každou lze ovládat samostatně. Stykové obvody těchto speciálních zobrazovacích jednotek zahrnují obvykle vlastní specializovaný grafický procesor, který zobrazení řídí.

Používáme-li monitor pro grafické zobrazení, narážíme na určité problémy s rozsahem obrazové paměti. Např. pro běžný monitor s rozsahem 348×720 bodů využitý pouze abecedněčíslicově postačí v nejjednodušším případě obrazová paměť o rozsahu 2 000 slabik (jedna slabika na znak). Pokud chceme dovolit efektnější zobrazení (např. inverzní zobrazení, blikání apod.), přidáme např. ke každému znaku další slabiku, ve které zadáváme jeho atributy. Obrazová paměť bude mít rozsah 4 000 slabik. Pro zapamatování všech bodů stínítka však potřebujeme 250 560 bitů, tj. zhruba 32 KB obrazové paměti.

Ještě markantněji se tento problém projeví při barevném zobrazení. Na každý zobrazovaný bod je nutno uchovávat v obrazové paměti podíl jednotlivých barvotvorných složek a jas. V řadě případů jsou proto při

barevném grafickém zobrazení použity různé triky. Jedním takovým trikem je zmenšení počtu zobrazovaných elementů. V obrazové paměti neudržujeme informace pro každý bod stínítka, ale vždy pouze pro určitou skupinu sousedících bodů (tzv. pixel). Jiným trikem je zmenšení rozsahu informace potřebné pro jeden zobrazovaný element. Místo abychom si pamatovali barvotvorné informace pro každý element, předdefinujeme omezenou aktuální paletu barev, z níž pak vybíráme již "namíchané" barvy (viz obr. 2.36).



Obr. 2.36. Řešení zobrazovací jednotky počítače MZ-800

Vývoj zobrazovacích jednotek neustále pokračuje, zkoušejí se různé jiné fyzikální principy. Např. pro přenosné mikropočítače se často využívá zobrazovací jednotky z tzv. tekutých krystalů, která má velmi malou spotřebu energie. Pro zvýraznění obrazu se zde využívá změny optické polarizace krystalů. Jednou z novinek poslední doby je tzv. transparentní obrazovka, vytvořená rovněž na bázi tekutých krystalů. Obrazovka se využívá v kombinaci např. se zpětným projektorem, který prosvítí transparentní stínítko a přenese obraz na plátno.

2.8.2.2 *Spolupráce se zobrazovací jednotkou*

Spolupráce mikropočítače se zobrazovací jednotkou může probíhat dvojím způsobem. Zobrazovací jednotku je možno připojit k systému jako

terminál, kdy pomocí stykových obvodů vytvoříme komunikační kanál směrem od mikroprocesoru k zobrazovací jednotce. Po tomto kanálu posíláme kódy znaků, které se mají zobrazit. Některé kódové kombinace nereprezentují znaky, ale představují speciální povely pro zobrazovací jednotku – např. vymaž obrazovku, nastav kurzor do udané pozice, či přejdi do grafického režimu. *Obrazová paměť* je v tomto případě součástí zobrazovací jednotky a z hlediska procesoru představuje *vnější paměť*.

Jiná možnost je využít pro obrazovou paměť část *vnitřní operační paměti*. Stykové obvody zobrazovací jednotky pak zajišťují cyklické obnovování obrazu části operační paměti, obvykle přímým přístupem do paměti. Zobrazovací jednotka je bez vlastní paměti. Nejnáročnější grafické zobrazovací jednotky obvykle využívají zdvojeného přístupu do obrazové paměti, kdy je pro přenos obsahu obrazové paměti do zobrazovací jednotky vytvořena samostatná cesta mimo sběrnici systému.

2.8.2.3 *Tiskárny*

Tiskárny představují jedno z nejdůležitějších výstupních zařízení počítačů, neboť v řadě aplikací (např. příprava textů) je tištěný dokument hlavním produktem práce systému. Je známa celá řada typů tiskáren, založených na rozmanitých principech záznamu. Z hlediska možností výstupní kresby lze tiskárny rozdělit na *grafické* (dovolují vytvořit v podstatě libovolný motiv), *semigrafické* (výstupní kresba se skládá pouze z povolené sady motivů) a znakové (povolená sada motivů je omezena na znakové symboly). Podle způsobu vytváření kresby se tiskárny dělí na bodové (kresba je složena z jednotlivých bodů), nebo *konturové* (tisk se vytváří z celých obrazců). Technologický postup získání kontrastních ploch na tiskovém médiu klasifikuje tiskárny na úderové a bezúderové.

Tiskárna je poměrně nákladné a často rozměrné zařízení, což ovlivňuje spektrum typů, používaných u mikropočítačů. Nejčastěji se zde setkáme s tiskárnami mozaikovými a tiskárnami s typovým kolečkem. U starších typů mikropočítačů se často používal dálnopis (teletype), kumulující funkce zobrazovací jednotky, klávesnice i tiskárny.

Mozaikové tiskárny vytvářejí tištěný obrazec z jednotlivých bodů, s využitím schopností lidského oka vnímat vzniklou mozaiku jako symbol. Převážná většina z nich pracuje jako úderové, lze se však setkat i s tepelnými

či elektrostatickými tiskárnami, kdy jsou body mozaiky zvýrazněny na speciálním papíře ohřátím nebo elektrostatickým výbojem. Jejich nevýhodou je právě nutnost použití speciálního média, na rozdíl od úderových, kde se používá běžný papír. U úderových mozaikových tiskáren jsou body tištěny úderem přes barvicí pásku.

Tisková hlava je většinou konstruována jako sloupec několika jehel uspořádaných nad sebou. Nejlevnější tiskárny mají jehlu jednu, náročnější obvykle 5 až 9 a nejnáročnější až 24. Výjimečně jsou jehly uspořádány do bodového řádku, který se tiskne najednou či nadvakrát (s posuvem). Existují i mozaikové tiskárny, kde jsou jehly (citlivé na úder) nahrazeny hranou kladívka, proti němuž se otáčí lopatkový válec. Jedním stykem hrany kladívka s lopatkou válce je vytvořen jeden bod na tiskovém médiu.

Sloupcová tisková hlava se musí samozřejmě pohybovat ve směru kolmém na sloupec jehel, aby bylo možno pokrýt dvourozměrné tiskové médium postupným ražením sloupců.

Typická mozaiková tiskárna pracuje jako znaková, tj. přijímá přes stykové obvody kódy znaků, které transformuje přes znakový generátor na matice bodového rastru znaků (používané rastry bývají 5×7, 7×9, ale i 24×24 bodů na znak). Generátor znaků bývá realizován jako pevná paměť o šířce buňky odpovídající počtu jehel. Adresa se skládá z kódu znaku a pořadí tištěného sloupce. Někdy obsahuje tiskárna mimo pevný generátor znaků i paměť typu RWM, do níž lze nahrát rozmanité vlastní repertoáry znaků (tzv. download).

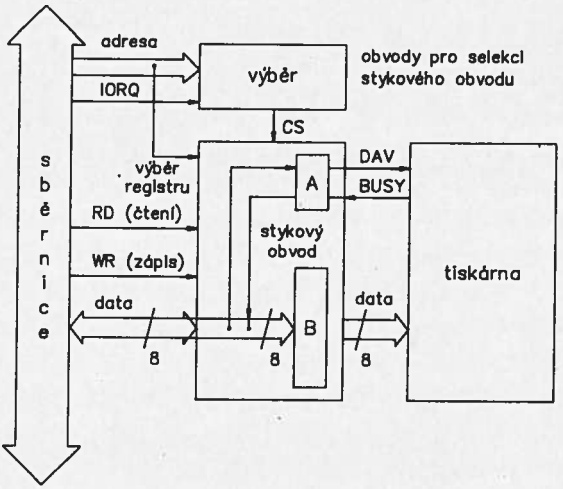
Náročnější mozaikové tiskárny dovolují práci ve dvou režimech – znakovém a grafickém. Ve znakovém režimu generuje tiskárna sloupce pro znaky sama (přenášejí se kódy znaků), v grafickém režimu se přenášejí přímo jednotlivé sloupcové kombinace. Nejnáročnější mozaikové tiskárny mimo to umožňují vícenásobný přetisk řádky s mikroposuvem, čímž je dosaženo překrytí a splynutí sousedních bodů mozaiky. Výsledný tisk je pak kvalitou srovnatelný s konturovým tiskem (tzv. letter quality).

Konturový způsob tisku používá tiskárna s typovým kolečkem (rozetou, kopretinou, sedmikráskou - pův. anglický termín je daisy wheel). Tisk se provádí úderem kladívka na některý z lístků typového kolečka, opatřených na obvodu konturami znaků. Typové kolečko se otáčí (to umožňuje výběr libovolného znaku) a současně spolu s kladívkem pohybuje v horizontálním směru. Počtem lístků je omezen repertoár znaků (obvykle 96), což může být

nevýhoda oproti mozaikové tiskárně. Na druhé straně jen nejkvalitnější mozaikové tiskárny produkují srovnatelný tisk. Určitou nevýhodou zmíněných typů tiskáren je poměrně pomalejší tisk (řádově stovky znaků za minutu).

Vývoj tiskáren však neustále pokračuje a existují již tiskárny, umožňující produkci řádově desítek až stovek stránek za minutu (tj. desítky až stovky tisíc znaků za minutu), použitelné i v mikropočítačových systémech. Především se jedná o *laserové tiskárny*, pracující na principu elektrofotografického tisku. Obraz tištěného dokumentu je vytvořen laserovým paprskem na fotocitlivém povrchu rotujícího bubnu, a poté vyvolán tónovačem a kontaktním způsobem přetištěn na papír. O něco pomalejší jsou tzv. *tryskové tiskárny* (ink jet), vytvářející obraz tryskajícím inkoustovým paprskem. Tryskovou tiskárnou lze mícháním různobarevných inkoustů docílit i barevného tisku.

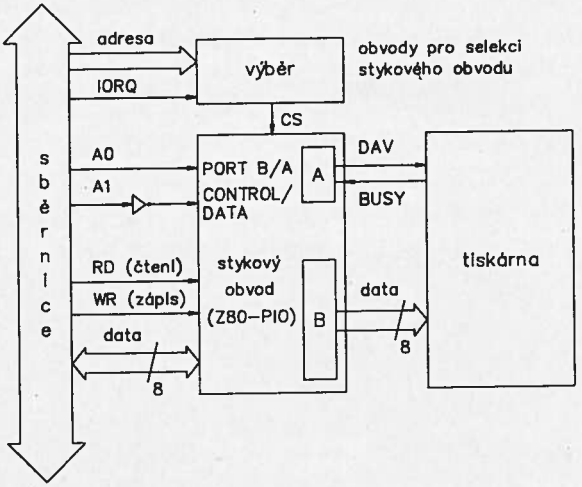
Spolupráce s tiskárnou nepřináší poměrně nic nového. Stykové obvody přijímají na straně mikropočítače kódy tištěných znaků a povely pro přenos na tiskárnu. Vnější rozhraní s tiskárnou může být provedeno paralelně nebo sériově, lze zde využít některého obecného rozhraní, kterým se budeme zabývat dále.



Obr. 2.37. Stykové obvody tiskárny počítače MZ-800

2.8.2.4 *Programové ovládání tiskárny*

Uvažujme pro ilustraci příklad jednoduchého ovladače tiskárny. Řešení stykových obvodů pro tiskárnu počítače MZ–800 je uvedeno na obr. 2.37. Rozhraní tiskárny se skládá z osmi datových vodičů a dvou řídicích signálů – vstupní signál pro tiskárnu, oznamující přítomnost dat na datových vodičích (DAV – DAta Valid), a výstupní signál, kterým tiskárna oznamuje, že právě tiskne a není schopna přijímat data (BUSY).



Obr. 2.38. Zapojení stykových obvodů tiskárny MZ–800

Na straně mikropočítače máme k dispozici běžnou sběrnici. Jako stykový obvod je v počítači MZ–800 použit univerzální programovatelný obvod Z80–PIO (Paralel Input/Output Interface). Zapojení stykových obvodů je uvedeno na obr. 2.38, kde jsou na straně rozhraní tiskárny z obvodu PIO využity dvě brány. Brána B je výstupní 8bitová a slouží pro připojení datových vodičů. Druhá brána A je rovněž osmibitová, přičemž prvých 6 bitů (0 až 5) je vstupních a vyšší 2 bity (6, 7) jsou výstupní. Signál DAV připojíme k bitu 7, signál BUSY k bitu 0.

U obvodu PIO dosáhneme příslušné konfigurace bran zápisem několika řídicích slabik do řídicích registrů.Posloupností povelů 0FFH a 3FH

zapsaných do řídicího registru brány A dosáhneme, aby brána A pracovala jako obousměrná, kde slabika 3FH udává vstupní bity brány. Zápisem povelu 0FH do řídicího registru brány B ji nastavíme jako výstupní. Protože obvod PIO obsahuje logiku pro napojení na přerušovací systém, musíme u obou bran zabránit generování žádosti o přerušení zápisem povelu 07H do datových registrů bran.

Na straně mikropočítače připojíme obvod PIO na datovou sběrnici. Dále připojíme signály RD a WR řídicí sběrnice na vstupy obvodu PIO, čímž umožníme řízení toku dat po datových vodičích. I/O paměť obvodu PIO tvoří 4 buňky – řídicí a datové registry bran A a B. Jejich selekce se provádí vstupními signály PORT B/A SEL a CONTROL/DATA SEL obvodu PIO dle tabulky:

CONTROL/DATA SEL	PORT B/A SEL	Význam
0	0	datový registr brány A
0	1	datový registr brány B
1	0	řídicí registr brány A
1	1	řídicí registr brány B

Celková aktivace obvodu se provádí vstupním signálem CS (Chip Select). Použitý mikroprocesor Z80 má oddělený I/O adresní prostor o celkovém možném rozsahu 256 I/O adres. Buňkám obvodu PIO přiřadíme I/O adresy dle tabulky.

I/O adresa	I/O buňka
0FCH	řídicí registr brány A
0FDH	řídicí registr brány B
0FEH	datový registr brány A
0FFH	datový registr brány B

Připojení na adresní sběrnici provedeme tak, že adresní vodič A0 připojíme přímo na vstup PORT B/A SEL obvodu PIO a adresní vodič A1 připojíme invertovaný na vstup CONTROL/DATA SEL. Vodiče A2 až A7 adresní sběrnice přivedeme spolu se signálem IORQ na vstup výběrového

obvodu, který v případě, že na adresní sběrnici je kombinace 00000000111111XX, tj. některá z adres 0FCH až 0FFH, a je nastaven signál IORQ, aktivuje obvod PIO signálem CS. Tím je připraveno technické řešení pro vytvoření ovladače tiskárny s aktivním čekáním.

```
; KOSTRA JEDNODUCHÉHO OVLADAČE TISKÁRNY
; DEFINICE SYMBOLICKÝCH KONSTANT
PIOAC   EQU 0FCH      ; I/O adresa řídicího registru brány A
PIOBC   EQU 0FDH      ; I/O adresa řídicího registru brány B
PIOAD   EQU 0FEH      ; I/0 adresa datového registru brány A
PIOBD   EQU 0FFH      ; I/0 adresa datového registru brány B
MODE0   EQU 00FH      ; mód 0 brány (vstupně/výstupní)
MODE3   EQU 0FFH      ; mód 3 brány (výstupní)
INITA   EQU 03FH      ; konfigurační povel pro bránu A
INTDI   EQU 007H      ; povel pro maskování přerušení
DAVON   EQU 080H      ; povel pro nastavení signálu DAV
DAVOF   EQU 000H      ; povel pro zrušení signálu DAV
MBUSY   EQU 001H      ; maska pro signál BUSY
; INICIALIZAČNÍ ČÁST (služba Init)
INIT: DI              ; zakaž přerušení procesoru
      LD   A, MODE3   ; konfigurace brány A obvodu PIO
      OUT (PIOAC), A
      LD   A, INITA   ; nastavení vstupních
      OUT (PIOAC), A  ; a výstupních vodičů
      LD   A,MODE0    ; konfigurace brány B obvodu PIO
      OUT (PIOBC), A
      LD   A,INTDI    ; maskuj přerušení od obvodu PIO
      OUT (PIOAD), A
      OUT (PIOBD), A
      EI              ; povol přerušení procesoru
      RET
; TEST STAVU TISKÁRNY
; PARAMETRY:
; vstup:   -
; výstup:  registr A = 0 .... tiskárna volná
;          registr A = 255 ... tiskárna obsazena
```

```
LPTST:  IN   A, (PIOAD)     ; sejmi obsah brány A
        AND MBUSY           ; vyber signál BUSY
        LD   A, 0           ; je-li volná, vrať 0
        RET  Z
        DEC  A              ; jinak vrať 255
        RET
; AKČNÍ ČÁST (služba TISK ZNAKU)
; PARAMETRY:
; vstup:  registr C – tištěný znak
TISK:   CALL LPTST          ; čekej na uvolnění tiskárny
        OR   A
        JR   NZ,TISK
        LD   A, C           ; zapiš znak do dat.reg. brány B
        OUT (PIOBD),A
        LD   A, DAVON       ; vydej povel pro tisk
        OUT (PIOAD), A
TISK1:  CALL LPTST
        OR   A
        JR   NZ, TISK1      ; čekej na dokončení tisku
        LD   A, DAVOF       ; nuluj signál DAV
        OUT (PIOAD), A
        RET
```

Současná tiskárna obsahuje obvykle vlastní mikropočítač, který řídí vlastní tisk i komunikaci se stykovými obvody. Volba různých režimů tisku se provádí programově, vysláním speciálních řídicích posloupností (obvykle indikovaných znakem ESC – tzv. escape sekvencí).

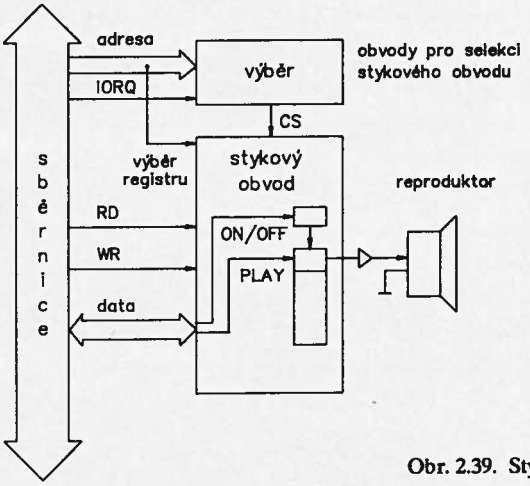
2.8.2.5 *Zvukový výstup*

Celá řada mikropočítačů je vybavena generátory zvuku. Jeden z hlavních důvodů jejich přítomnosti je využití zvuku v počítačových hrách. Proto je také nezřídka vybavení domácích počítačů pro generování zvuku bohatší než u vyšších tříd osobních počítačů.

Prostý zvukový výstup se skládá z elektroakustického měniče (reproduktoru) a stykových obvodů, umožňujících vyslat do měniče

napěťový puls. Obvykle se zvukový výstup ovládá dvěma signály – stykové obvody rozeznávají dva samostatné povely (obr. 2.39):
– zapni/vypni stykový obvod zvukového výstupu,
– připoj/odpoj napětí od reproduktoru.

Po zapnutí zvukového výstupu lze střídáním povelu připoj a odpoj v určité frekvenci vytvořit odpovídající zvukový signál, např. podaří-li se připojit a odpojit reproduktor 440krát za sekundu, měli bychom slyšet komorní A.



Obr. 2.39. Stykové obvody pro zvukový výstup počítače MZ–800

Jednoduché zapojení zvukového výstupu má dvě základní nevýhody – zaměstnává plně procesor měřením časových úseků a generovaný zvuk je závislý na rychlosti provádění instrukcí (hodinové frekvenci procesoru). Výhodnější je proto použití programovatelného časovače pro generování stabilní frekvence. Nastavením požadované vlnové délky a spuštěním časovače lze pak dosáhnout stabilní generování pulsů odpovídající frekvence, které přivedeme na spínací vstup reproduktoru. Spuštění a vypnutí zvuku zajistíme povelem zapni/vypni stykový obvod.

U domácích počítačů bývá často použit složitější stykový obvod, který dovoluje generování až čtyř nezávislých hlasů (zvuků), včetně ovládání hlasitosti jednotlivých zvuků (součástí povelu je určení hlasitosti). Stykové obvody mohou ovládat buď zabudovaný reproduktor, nebo vnější akustickou

soustavu, příp. může být výstup namodulován k výstupnímu televiznímu signálu pro televizní zobrazovací jednotku.

Moderní současné osobní počítače bývají vybaveny často speciálními stykovými obvody pro komunikaci se zařízeními pro digitální úpravu zvuku – tzv. norma MIDI (Music Interchange Digital Interface).

2.8.2.6 *Souřadnicové zapisovače*

Grafické výstupní možnosti mikropočítače bývají někdy doplněny tzv. souřadnicovým zapisovačem (plotter). Jedná se o zařízení, které umí běžnými kreslicími prostředky vytvářet základní grafické motivy (úsečky, kružnice, kruhové oblouky, text apod.), zadané charakteristikami v jistém souřadném systému. Kresba se provádí buď dvěma kolmými posuvy pisátka, nebo kombinací pohybu pisátka a papíru. Připojení i ovládání je obdobné jako u tiskárny.

2.8.3 Vnější paměti

V předchozích odstavcích jsme probrali přídavná zařízení, která umožňovala buď pouze vstup informace do mikropočítače, nebo pouze výstup informace z mikropočítače. Neméně důležitá jsou však zařízení, připouštějící obousměrný přenos. Do této kategorie se řadí jednak komunikační zařízení pro styk mikropočítače s okolím a zejména vnější paměti. Jedním z nejzávažnějších limitujících faktorů použitelnosti mikropočítačového systému je právě kapacita a rychlost vnějších pamětí.

Vnější paměť představuje přídavné zařízení, které uchovává přijatou informaci s využitím určitého fyzikálního principu pro pozdější opětovné použití. Vzhledem k postavení vnějších pamětí v hierarchii paměťových systémů mikropočítače se předpokládá záznam velkého množství informace při delší době přístupu. Nevýhoda delší doby přístupu je částečně kompenzována zvětšením rozsahu přenášené informace. Záznam do vnější paměti se proto neprovádí po jednotlivých informačních jednotkách mikropočítače (slabika, slovo), ale po větších blocích. Rovněž výběr informace z vnější paměti je možný pouze po blocích, což jsou z hlediska vnější paměti minimální identifikovatelné celky.

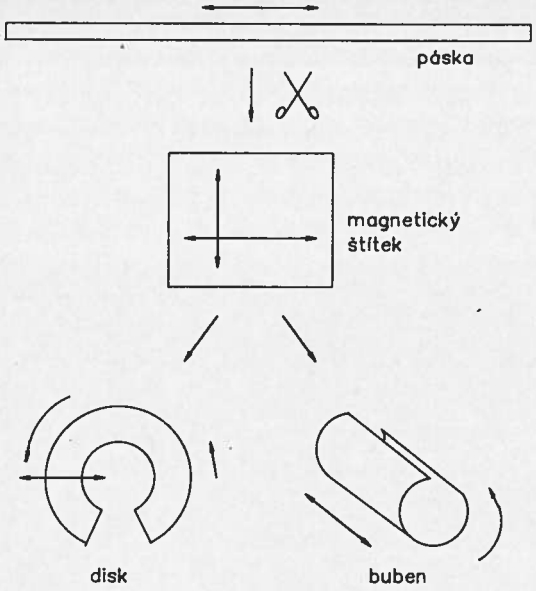
Pro identifikaci místa uložení informace (bloku) se používá opět identifikačních čísel – adres. *Adresa* ve vnější paměti se však často skládá z několika složek, pokud je vnější paměť jistým způsobem strukturována, např. disková adresa se skládá z čísla válce (stopy), čísla povrchu a čísla sektoru. Celou situaci lze chápat tak, že vnější paměť je vícerozměrná a adresu tvoří souřadnice v odpovídajícím prostoru. Poznamenejme, že stejný trik bývá někdy použit i u vnitřní operační paměti.

Záznamové médium vnějších pamětí zatím nejčastěji tvoří magnetická vrstva, ale používají se i jiné pricipy záznamu. Velký rozvoj prodělává v současné době např. optický způsob záznamu (např. pomocí laseru), kde je možno dosáhnout mnohem vyšší hustoty záznamu. Oproti magnetickému médiu má optický záznam zatím nevýhodu v nesnadné realizaci přepisu uložené informace – optický záznam je často buď permanentní (vytvořený během výroby), nebo je možno informaci do média zaznamenat pouze jednou (tzv. WORM média – Write Once Read Multiple).

V nejjednodušším případě je záznamové médium jednorozměrné – typickým představitelem je kazetová magnetická páska, kde se pohybuje médium, nebo jednorozměrná magnetická bublinová paměť, kde se pohybuje informace v nekonečné smyčce.

Jednorozměrné médium má výhodu jednoduché technické realizace, na druhé straně však principiálně způsobuje poměrně dlouhou dobu přístupu k uložené informaci. Situaci lze řešit pomyslným rozstříháním jednorozměrného média na úseky stejné délky a slepením do dvourozměrného média. Pohyb ve druhém rozměru pak fakticky simuluje velmi rychlý pohyb na jednorozměrném médiu. Technický problém však představuje levná realizace dvou kolmých lineárních posuvů. Výhodnější je nahradit jeden z lineárních posuvů otáčením. Dvourozměrné médium pak musíme zacyklit slepením dvou stran – protilehlých či sousedních (obr. 2.40). Vznikne tak médium ve tvaru válce (bubnu) či disku. Nevýhodou válce je příliš velký objem a obtížná realizace více povrchů (třetího rozměru). Malý objem a snadná realizace více povrchů jsou naopak výhody diskového média, a přestože má disk oproti válci nevýhodu v nestejnoměrné hustotě záznamu, ostatní výhody převažují.

V současné době se u mikropočítačů téměř výhradně používají disková media s různými způsoby záznamu, příp. kazetopásková magnetická média. Zmíníme se proto o těchto typech vnějších pamětí podrobněji.

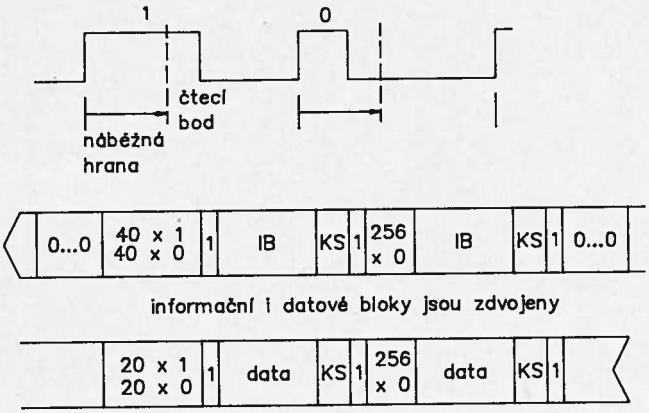


Obr. 2.40. Zacyklení dvourozměrného média

2.8.3.1 *Kazetopáskové magnetické paměti*

Cenově optimální řešení vnější paměti pro domácí počítač je *kazetopásková vnější paměť*. Lze zde využít komerčně užívaných kazetových magnetofonů nebo magnetofonů speciálních (kvalitnějších). Na rozdíl od běžných magnetických páskových pamětí, používaných u větších počítačů, kde se zaznamenává paralelně několik stop (nejčastěji 9, tj. 8 stop informačních a jedna kontrolní), je záznam na kazetový pásek sériový v jediné stopě. Zhruba řečeno, spočívá princip záznamu v přidělení různých frekvencí různým logickým úrovním datového signálu a "nazpíváním" logických not na pásek. Podobného efektu lze dosáhnout fázovou modulací nosného kmitočtu, kdy je informace zaznamenána pomocí změn fáze (např. změnou polarity signálu). Záznam se provádí v blocích opatřených kontrolními informacemi (např. kontrolním součtem). Na neštěstí pro uživatele se používá celé řady způsobů záznamu (způsob modulace, délka bloku, kontrolní informace atd.). Jedním z důvodů je i fakt, že výrobci domácích počítačů takto nutí uživatele využívat firemní zařízení.

Na obr. 2.41 je uveden příklad možného způsobu záznamu na kazetovou pásku použitý v počítači MZ–800. Jednotlivým logickým hodnotám jsou přiřazeny různé frekvence změn. Logické hodnotě nula odpovídá puls délky přibližně 200 μs, logickou hodnotu jedna reprezentuje puls délky přibližně 400 μs. čtecí logika odměřuje čas od počátku pulsu a sejme informaci v čase přibližně 300 μs od vzestupné hrany. Trvá-li puls, jedná se o hodnotu jedna, jinak o hodnotu nula. Informační bloky jsou uvozeny posloupností nul pro nastavení časové synchronizace. Z důvodů spolehlivosti čtení je informace na pásku zaznamenána dvakrát.



Obr. 2.41. Způsob záznamu na kazetovou pásku pro MZ–800
(KS – kontrolní součet – 2 slabiky, IB – informační blok – 128 slabik)

2.8.3.2 *Spolupráce s kazetovou pamětí*

Spolupráce s kazetovou pamětí u domácích počítačů probíhá na poměrně velmi nízké úrovni. Stykové obvody pro spolupráci obvykle zajišťují pouze převod signálu mezi elektrickou úrovní v počítači a tvarem signálu v magnetofonu. Z hlediska programového ovládání vytvářejí stykové obvody jedinou linku, po níž se přenáší informace v sériovém tvaru po jednotlivých bitech. Obslužné programy se musí přizpůsobit rychlosti toku dat v magnetofonu a zodpovídají i za vytváření a zpracování informačních bloků, včetně kontrolní informace. V lepším případě

umožňují stykové obvody i řízení posuvu pásky, obvykle však pouze na úrovni spuštění či zastavení posuvu.

Hustota záznamu informace na běžné kazetové pásce (při zachování dostatečné spolehlivosti záznamu) je z hlediska nároků mikropočítače nedostatečná. Přenos informace o mohutnosti odpovídající rozsahu běžné operační paměti trvá řádově minuty, nehledě na čas, potřebný k vyhledání informace na pásce. Určitým příslibem do budoucna jsou magnetofony s rotujícími hlavami a šikmým záznamem, kde se rychlou rotací hlav dosahuje jednak lepšího využití plochy pásky a zejména značného zvýšení relativní rychlosti posuvu hlav vůči pásce.

Při velké relativní rychlosti posuvu lze ve stejném časovém úseku zaznamenat vyšší množství informace, při zachování stejné (magnetické) vzdálenosti jednotlivých informačních bitů. Vysoké relativní rychlosti posuvu se přitom nedosahuje zvýšením rychlosti posuvu pásky, ale pouze rychlostí otáčení hlav. Tato koncepce dovoluje zvýšit hustotu záznamu na míru, která by mohla být pro osobní počítače postačující. Princip rotujících hlav však dovoluje ještě další vylepšení proti běžné pásce, neboť při zvýšení rychlosti posuvu pásky a současném snížení rychlosti otáčení hlav (v určitých mezích), zůstává vzájemná relativní rychlost stejná. Jinými slovy, pásku lze číst i při rychloposuvu (přetáčení), byť s menší spolehlivostí vzhledem k běžnému režimu, ale dostatečnou pro indikaci speciálních značek – hranic bloků či celých souborů. Tím se částečně snižuje i problém rychlého vyhledání informace, uložené na pásce.

Systém s rotujícími hlavami by mohl (zejména cenově) konkurovat i diskovým pamětem, zejména jako levné médium pro archivaci. Obdobný způsob záznamu se používá u videomagnetofonu, a proto je využití videomagnetofonu jako přídavného zařízení mikropočítače velmi perspektivní. Současně je pak možno realizovat vytváření obrazu programovými prostředky.

2.8.3.3 *Disketové paměti*

Velkým přínosem pro vznik mikropočítačových systémů byl fakt, že v téže době, kdy byly k dispozici komponenty pro stavbu procesoru s vnitřní pamětí mikropočítačového systému, objevují se na trhu tzv. *disketové paměti* (paměti s pružnými disky). Teprve kombinace mikroprocesoru, polovo-

dičové vnitřní paměti a disketové paměti představuje dostatečný základ pro vznik skutečně použitelného miniaturního výpočetního systému.

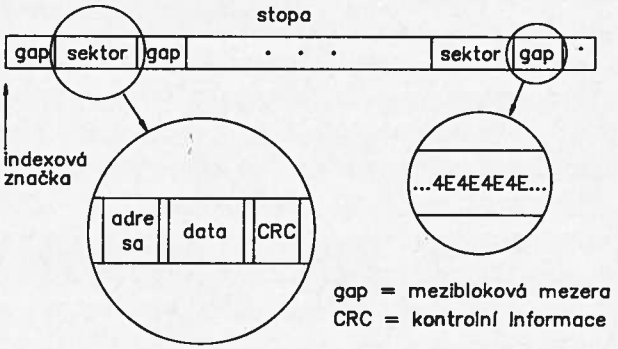
Disketová paměť byla vyvinuta původně pro zcela jiné účely – jako levná trvalá paměť (paměť typu ROM) pro větší výpočetní systémy. Její návrh předložil v roce 1967 D. L. Noble u firmy IBM. Dosažené parametry, zejména rychlost přístupu, spolehlivost a malé výrobní náklady, však předurčily disketovou paměť pro širší použití. V roce 1971 uvedla firma IBM na trh model 33 FD, který se stal v podstatě průmyslovým standardem (IBM 3740) a představoval ideální médium pro mikropočítačové systémy. V roce 1973 vyvinul J. Torode řadič paměti s pružným diskem, umožňující spolupráci s mikropočítačem, a otevřel tak disketové paměti bránu do světa mikropočítačů.

Základem disketové paměti je pružný kotouč – disk, pokrytý magnetickou vrstvou. Disk se otáčí (obvykle rychlostí 360 ot/min) mezi kluznými a čisticími vložkami uvnitř čtvercového polotuhého obalu. Na rozdíl od běžného disku je pružný disk čten a zapisován kulovou magnetickou hlavou, která se dotýká povrchu média (to je možné vzhledem k malé rychlosti otáčení a pružnosti média). Hlava se může pohybovat radiálně pomocí krokového motoru, čímž se mění mezikruží, ke kterému má hlava na médiu přístup.

Informace je na pružném disku zaznamenávána v kruhových soustředných stopách. Termín *stopa* se u disketových pamětí vžil vzhledem k tomu, že u původních médií se využíval pouze jediný povrch – ačkoliv již existoval adekvátní pojem *válec* (cylindr). Např. standardní *osmipalcový pružný disk* (IBM 3740) obsahuje 77 stop (stopy 0 až 76). Každá stopa je rozdělena na úseky určité délky – tzv. *sektory*. Rozdělení na sektory může být pevné, vyznačené mechanicky pomocí dírek poblíž vnitřního otvoru média a indikováno fotocitlivým prvkem. Takto strukturovaný disk se nazývá *pevně sektorovaný disk* (hard sectored).

U mikropočítačů je však mnohem běžnější jiný způsob dělení stopy na sektory, kterému říkáme *variabilně sektorovaný disk* (soft sectored). V tomto případě je mechanicky indikován dírkou pouze začátek stopy. Za vytvoření příslušné sektorové struktury na stopě disku zodpovídá řadič diskety, příp. ve spolupráci s programem. Vytváření sektorové struktury na stopě nazýváme *formátování stopy*. Během formátování se na stopě vytvoří speciální značky, které vymezují hranice jednotlivých sektorů. Záznam je opět sériový, možný formát stopy je uveden na obr. 2.42.

Každý sektor se skládá z identifikační a datové části. Obě části jsou opatřeny kontrolní informací, obvykle tzv. CRC (Cyclical Redundancy Check), což je informace dovolující detekci chyb v záznamu. V identifikační části sektoru je zaznamenána fyzická adresa (číslo stopy, číslo povrchu a číslo sektoru) a délka datové části sektoru. Účelem záznamu adresy do sektoru je možnost přesné identifikace sektoru, bez ohledu na jeho umístění na stopě. Sektory mohou být na stopě přeházeny, což je někdy výhodné při zpracování, neboť při programové obsluze přechodu na další sektor nám může jeho začátek uniknout, a nezbývá než vyčkat celou otáčku média. Pokud je následující sektor uložen fyzicky až později, není toto zdržení nutné.



Obr. 2.42. Formát stopy na disketě

Např. stopa standardní osmipalcové diskety (formátu IBM 3740) je rozdělena na 26 sektorů stejné délky (sektory 1 až 26), které obsahují služební informace a vlastní data (128 slabik). Celková fyzická kapacita standardní osmipalcové diskety je tedy 77×26 = 2 002 sektorů, tj. 77×26×128 = = 256 256 B = 256 KB. Skutečně použitelná kapacita diskety je o něco menší, vzhledem k nutné režii, potřebné pro organizaci uložení informace na médiu.

Standardní osmipalcová disketa formátu IBM 3740 využívá pouze jednu stranu pružného disku, v jednoduché hustotě záznamu. Existují jiné formáty osmipalcových disket, kdy se využitím úspornějšího kódování informace dosáhne dvojnásobné hustoty záznamu a příp. se používají obě strany média. Lze tak dosáhnout informační kapacity okolo 1 MB, ale není zde

žádný obecný standard obdobný formátu IBM 3740, který je naproti tomu použitelný na téměř každém 8bitovém mikropočítači.

Disketová paměť je předmětem neustálého vývoje. U 16bitových mikropočítačů se typicky používají zejména *pětačtvrtpalcové diskety* (minidiskety). Např. pro třídu 16bitových osobních mikropočítačů, srovnatelných s mikropočítačem IBM PC, je charakteristické použití několika standardních formátů pětačtvrtpalcových disket. Minidisketa je rozdělena na 40 stop, formátovaných variabilně do osmi či devíti sektorů délky 512 B. Využívá se buď jeden, nebo oba povrchy pružného disku. Tomu odpovídají kapacity 160, 180, 320 a 360 KB. Často se lze setkat s tzv. *kvadratickou hustotou*, kdy se zúžením záznamu dosahuje zvýšení počtu stop na 80, formátovaných do 9 či 15 sektorů; získaná kapacita je tedy 720 KB, příp. 1,2 MB.

Pro 32bitové mikropočítače je téměř standardní použití *třiapůlpalcových disket* (mikrodisket), které jsou, na rozdíl od větších disket, uzavřeny v plastikovém pouzdře a opatřeny výsuvnými dvířky pro přístup čtecích a záznamových hlav. Mikrodisketa je proto velmi snadno přenosná a dostatečně chráněná proti poškození. V extrémních případech bylo dosaženo u mikrodisketových pamětí kapacity až kolem 10 MB, běžně se používají kapacity okolo 1 MB.

Disketová paměť má v mikropočítačovém systému tři základní funkce
– *paměť služebních programů* (operační systém, překladače, editory atd.),
– *pracovní paměť* pro ukládání informace, která se nevejde do operační paměti,
– *archivní médium* pro ukládání uživatelských programů a datových souborů
Přestože může dobře posloužit pro všechny tyto účely, je její koncepce vhodná zejména pro archivaci. Prvé dvě funkce by stejně dobře mohlo plnit zařízení, u nějž by nebylo nutné používat výměnné médium. Zde někde lze hledat prvotní ideu použití tzv. *pevných disků* (hard disk) v mikropočítačovém systému.

2.8.3.4 *Pevný disk*

Pevný disk má, na rozdíl od diskety, otočné médium pevně zabudováno do mechaniky. To umožňuje mnohem přesnější mechanické provedení, takže ačkoli se magnetické hlavičky záznamového povrchu

nedotýkají, lze je udržet ve velmi malé vzdálenosti. Celý disk lze dokonce hermeticky uzavřít, vyloučit tak nečistotu a prach a ještě více přiblížit hlavy k magnetickému povrchu (okolo 0,4 μm, tj. asi 1/200 tloušťky lidského vlasu, proti 0,8 μm u běžných pevných disků). Blízkost hlavy a magnetického povrchu dovoluje velkou hustotu záznamu (a tedy i velkou kapacitu disku – řádově 10^6 až 10^8 slabik). Naproti tomu fakt, že se hlavy povrchu nedotýkají, dovoluje velké rychlosti otáčení disku (3 600 ot/min), a tím krátkou dobu přístupu k libovolné informaci (přibližně desítky ms, tj. asi desetkrát až stokrát méně, než u diskety). Aby se zamezilo kolizi hlavy a povrchu a snížilo tření, je povrch disku opatřen velmi tenkou kluznou vrstvou. Pevným diskům této konstrukce se někdy říká "Winchester" disk podle prvního modelu IBM 3340, který měl mít dva disky o kapacitě 30 MB, což připomínalo známou pušku winchestrovku 30–30.

Začátkem osmdesátých let se pevné disky staly běžným vybavením mikropočítačů (např. IBM PC–XT) a značně tak zvyšily výpočetní kapacitu těchto systémů. Další generace těchto disků dovolují dokonce výměnné kazety, obsahující kromě disku i hlavičky. Přesto zůstávají i diskety běžným vybavením mikropočítače, neboť velké kapacity pevného disků s sebou nesou nebezpečí současné ztráty velkého množství dat při poruše média, a tím nutnost archivace např. na speciálních magnetických páskách (tzv. streamer tape). Označení vyplývá ze způsobu záznamu na pásku, který se provádí proudově bez zastavování. Pro tyto účely lze také využít výstup na videomagnetofon.

2.8.3.5 *RAM – disk*

Paměťové schopnosti mikropočítačů lze také rozšiřovat tzv. RAM–diskem, který se z hlediska operací chová stejně jako obvyklá disková paměť, ale je realizován elektronicky – pomocí stejných obvodů, jako vnitřní operační paměť (proto RAM-disk). Umožnila to snižující se cena polovodičových pamětí. RAM-disk s sebou přináší velké výhody:
– velké zvýšení paměťové kapacity bez ohledu na adresní prostor použitého mikroprocesoru (např. běžně je užíván RAM-disk s kapacitou 256 KB u 8bitových mikroprocesorů, které mají paměťový prostor jen 64 KB);

– velmi rychlý přístup k informaci (řádově tisíckrát rychlejší než u pružného disku); nahrajeme-li do RAM-disku všechny programy, odpadá zcela přenos programů z disku do paměti;
– fakt, že se RAM-disk chová jako běžný disk, dovoluje použít beze změny veškeré programové vybavení.

Existuje ale jedna nevýhoda, která nutí uživatele RAM-disku k opatrnosti – totiž ztráta informace při výpadku napájení nebo při vypnutí systému. V tomto směru se RAM-disk podobá pevnému disku – nesmíme zapomenout na archivaci dat.

2.8.4 Vnější rozhraní a komunikační linky

Většina osobních počítačů je vybavena technickými prostředky, které souhrnně nazýváme *vnější rozhraní*. Vnější rozhraní může být využito jak pro spolupráci s jinými systémy, tak i pro přenos dat mezi komponentami systému, např. mezi stykovými obvody v mikropočítači a přídavným zařízením. Vnější rozhraní může být koncipováno jako specializované, nebo do jisté míry univerzální (používáme pak také označení vnější sběrnice).

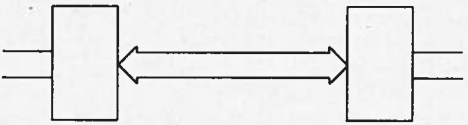
Komunikace na vnějším rozhraní může probíhat po základních informačních jednotkách (rozumí se z hlediska mikropočítače, tj. např. po slabikách), nebo po blocích pevné či proměnné délky. Přenos po základních informačních jednotkách je vhodný pro pomalá zařízení, kde je interval mezi přenosem dvou jednotek podstatně delší než doba potřebná k navázání spojení a provedení přenosu. Přenos po blocích pevné délky je účelný zejména u zařízení, která s pevnou délkou bloků pracují. Nejpružnější, ale současně nejnáročnější na realizaci, je přenos po blocích proměnné délky.

Vlastní přenosová cesta může být realizována sadou paralelních vodičů – mluvíme o *paralelním přenosu* (obr. 2.43), který je vhodný zejména pro rychlá zařízení na krátké vzdálenosti. Typická paralelní přenosová cesta používaná u mikropočítačů má šířku 8 informačních vodičů a lze po ní tedy přenášet vždy jednu slabiku.

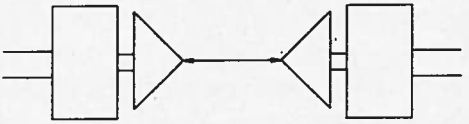
Pro pomalejší zařízení a větší vzdálenosti je z hlediska ceny přenosové cesty a spolehlivosti výhodnější sériový přenos informace po užším přenosovém médiu (obr. 2.44). Do přenosové cesty je pak nutno zahrnout serializaci dat a zpětný převod. Převod na sériový tvar lze řešit programově, výhodnější je však technická realizace ve stykových obvodech. Chceme-li využít pro

přenos standardních telekomunikačních linek, je dále třeba vložit do přenosové cesty tzv. *modem* (modulátor/demodulátor), který upraví signál z vnějšího rozhraní mikropočítače do potřebného tvaru (obr. 2.45).

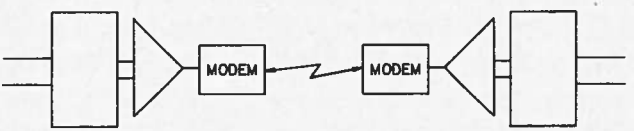
Během přenosu se vysílač a přijímač musí vzájemně přizpůsobovat. Rozeznáváme proto tzv. synchronní nebo asynchronní přenos. *Synchronní přenos* se vyznačuje tím, že činnost přijímače a vysílače je synchronizována zvláštním signálem. Tento synchronizační signál může být přenášen po samostatné cestě, což však vyžaduje zvýšení počtu přenosových vodičů. Výhodnější je proto použít takového kódování informace, kdy je synchronizační signál začleněn do datového signálu a přenášen po stejné přenosové cestě.



Obr. 2.43. Paralelní přenosová cesta



Obr. 2.44. Sériová přenosová cesta



Obr. 2.45. Komunikace po telefonních linkách

Při *asynchronním přenosu* se synchronizační signál nepřenáší, předpokládá se, že jak vysílač, tak přijímač jsou v rámci jednoho přenosu dostatečně synchronní. To lze zajistit samozřejmě pouze při přenosu dostatečně krátkých celků, obvykle pouze několika bitů. Naproti tomu synchronní způsob komunikace se používá i pro přenos větších celků. Z toho též plyne oblast použití obou způsobů, kdy synchronní způsob komunikace využíváme při spolupráci podobně rychlých

zařízení, zatímco asynchronní způsob komunikace je vhodný pro zařízení s různou rychlostí.

Synchronní způsob komunikace obvykle předpokládá přenos po blocích pevné nebo proměnné délky, zabezpečených na této úrovni kontrolními informacemi (tzv. rámce). *Rámce* mohou být orientovány znakově (přenášejí se posloupnosti znaků) nebo bitově (přenáší se posloupnost bitů). Na obr. 2.46 jsou znázorněny příklady možných formátů zpráv pro sériový synchronní přenos. Je-li synchronizační signál přenášen po datové cestě, jsou úseky před přenosem rámců vyplněny zvláštními znaky, kvůli navázání a udržení synchronizace.

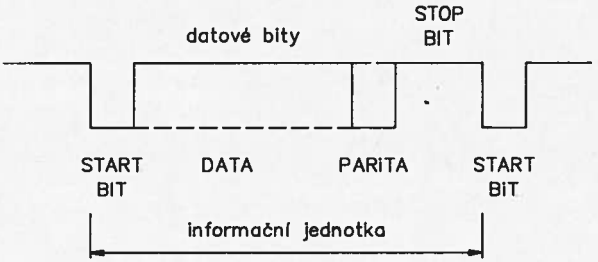
Obr. 2.46. Formát informace při synchronním sériovém přenosu

Asynchronní způsob komunikace naproti tomu předpokládá klid na lince mezi jednotlivými přenosy. Prvá změna způsobí zasynchronizování přijímače (tzv. START bit), který od této změny odměřuje dohodnuté časové intervaly, v nichž přicházejí informační signály dat. Pro zabezpečení přenosu se zde někdy používá parita (lichá nebo sudá), na rozdíl od synchronního přenosu, kde by paritní bity zbytečně prodlužovaly přenášený rámec, zabezpečený jako celek jinými metodami (obvykle pomocí CRC). Dohodnutý počet informačních bitů (5 až 8) je pak u asynchronní komunikace doplněn přenosem parity, po níž následuje povinný klid na lince (tzv. STOP bity). Charakteristický formát zprávy při asynchronním sériovém přenosu je znázorněn na obr. 2.47. Používané přenosové rychlosti jsou v rozsahu 15 až 19 200 bitů za sekundu, což odpovídá časovému odstupu bitů 66 až 0.05 ms.

Z hlediska operačních systémů jsou prostředky komunikace zajímavé zejména tehdy, chceme-li k počítači připojit nové zařízení, či se připojit k jinému počítači. Normalizované rozhraní přenosových cest tuto činnost značně usnadňuje. To je smyslem zavádění standardů pro vnější rozhraní. Pro připojování vnějších zařízení k mikropočítači existují různé normy; jako příklad může posloužit doporučení Mezinárodní elektrotechnické společnosti (International Electrotechnical Commision) IEC

Standard 625–1, známé u nás pod označením IMS–2 (Informační měřicí systémy 2. generace), které specifikuje mechanické (počet a význam vodičů, zapojení konektorů), elektrické (parametry elektrických signálů) i funkční (pravidla komunikace, formáty zpráv) rozhraní sběrnice pro připojování přídavných zařízení. Známé normy využívají různých způsobů komunikace. Přenos může probíhat paralelně (Centronics, IMS–2) nebo sériově (proudová smyčka, RS 232). V současné době jsou nejrozšířenější standardy IMS–2 pro paralelní připojení libovolného zařízení, Centronics pro paralelní připojení tiskáren, RS 232 pro sériové připojení rozmanitých zařízení včetně komunikačních modemů na kratší vzdálenost (do 15 m) a proudová smyčka pro sériové připojení vzdálenějších zařízení (do 1000 m).

Z hlediska ovládání je důležitý zejména rozdíl mezi jednotlivým a blokovým přenosem. V případě jednotlivého přenosu pracují stykové obvody samostatně, při blokovém přenosu spolupracují obvykle s obvody pro přímý přístup do paměti.

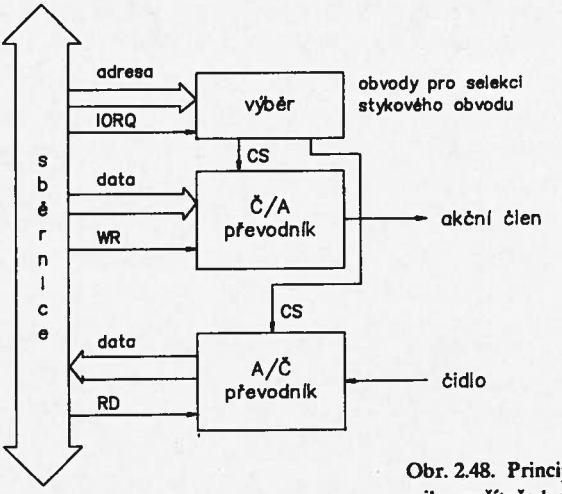


Obr. 2.47. Formát informace při asynchronním sériovém přenosu

2.8.5 Speciální přídavná zařízení

Mikropočítačový systém lze vybavit nepřeberným množstvím přídavných zařízení. Používáme-li jej jako řídicí prvek, často využíváme analogově-číslicových převodníků pro vstup údajů z různých čidel, a naopak číslicově-analogových převodníků pro ovládání akčních prvků (obr. 2.48). Pro řízení v reálném čase, ale i pro jiné účely (obnova informace na obrazovce, přepínání paralelních procesů atd.) je nutné připojení časovače,

který odměřuje různé časové intervaly. Časovač získává údaje o čase buď dělením základní frekvence procesoru, nebo pracuje jako samostatný prvek, s vlastním krystalem, akumulátorem a pamětí s velmi nízkou spotřebou energie. Jednoduchý časovač může být připojen pouze jako vstupní zařízení, které signalizuje v pevných časových intervalech mikropočítači uplynutí



Obr. 2.48. Princip připojení mikropočítače k analogové soustavě

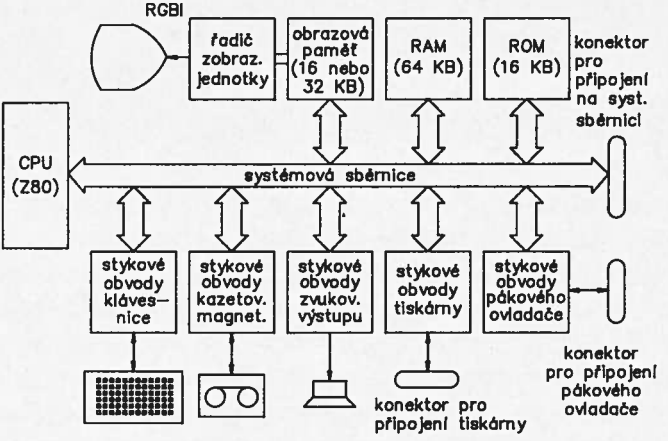
jistého časového intervalu. Tento signál pak obvykle bývá zpracován přes přerušovací systém. Chytřejší časovač lze nastavit (naprogramovat) pro hlášení různých časových intervalů a je tedy připojen současně jako výstupní zařízení, do kterého zapisujeme ovládací povely.

2.9 Příklad technického řešení mikropočítače

Na závěr kapitoly o technických prostředcích osobních počítačů uvedeme konkrétní technické řešení osobního počítače Sharp MZ–800. Smyslem prezentace by mělo být sjednocení znalostí získaných v této kapitole. V dalších kapitolách pak využijeme prezentované řešení při ilustraci vytváření operačního systému pro jednoduchý mikropočítač. Volba typu mikropočítače není podstatná a byla ovlivněna jednak dostupností

modelu MZ–800 na našem trhu, jednak jednoduchostí konstrukce. Současně se omlouváme předem všem znalcům mikropočítače MZ–800 za některá zjednodušení, kterých se kvůli přehlednosti v prezentaci dopustíme.

Osobní počítač Sharp MZ–800 je vybaven mikroprocesorem Z80A (frekvence hodin je 3.5469 MHz), dále 64 KB operační paměti RWM, 16 KB



Obr. 2.49. Sestava osobního počítače MZ–800

paměti ROM pro základní programové vybavení a až 32 KB obrazové paměti RWM. Konstrukční řešení je provedeno tak, že na základní desce je umístěn procesor i paměti. Na téže desce jsou zabudovány stykové obvody pro připojení klávesnice, monitoru, kazetopáskové jednotky, tiskárny, zvukového výstupu a dvou pákových ovladačů. Jako spojovací soustava je použita sběrnice (16 adresních vodičů, 8 datových vodičů). Část systémové sběrnice je vyvedena jako vnější rozhraní pro případné rozšíření systému. Celkovou základní sestavu ilustruje obr. 2.49.

3 Úvod do operačního systému CP/M

Operační systém CP/M (Control Program for Microcomputers) byl první operační systém vytvořený speciálně pro mikropočítače. Základní varianta systému je určena pro 8bitové mikropočítače osazené mikroprocesory Intel 8080, 8085 nebo Zilog Z80 a vybavené diskovými jednotkami (diskety, pevné disky apod.). Tato varianta bývá často označována též CP/M-80. Pro 16bitové mikropočítače existuje varianta CP/M-86, která pracuje na systémech vybavených mikroprocesory Intel 8086 či 8088 (příp. procesory, které umějí emulovat jejich kód, tj. např. Intel 80286). Existují rovněž varianty pro jiné procesory, např. pro 8bitové procesory Motorola 6800, Mostek 6502, či 16bitové procesory Motorola 68000 (CP/M-68K) nebo Zilog Z8000.

V současné době představuje systém CP/M pravděpodobně nejpopulárnější (a v podstatě standardní) diskový operační systém pro 8bitové mikropočítače. Odhaduje se, že systém CP/M je využíván asi na 80 % 8bitových osobních počítačů. Pro vyšší třídy mikropočítačů jsou typické jiné operační systémy, např. MS-DOS, či Unix. V této kapitole probereme stručně historii operačního systému CP/M a jeho verze, celkovou koncepci a architekturu. Dále se budeme zabývat požadavky na technické prostředí, ve kterém lze systém provozovat a způsoby jeho zavádění. Na závěr kapitoly shrneme základní pojmy, používané v terminologii systému CP/M.

3.1 Historie a verze operačního systému CP/M

Otcem CP/M je Gary Kildall, který jej v letech 1971 až 1973 navrhl jako součást implementace programovacího jazyka PL/M pro mikropočítač osazený procesorem Intel 8080 a nabídl jeho realizaci firmě Intel, u které

působil jako konzultant pro programové vybavení. Nescházelo mnoho a systém CP/M se mohl stát firemním systémem. V rámci reorganizací, způsobených rychlým vzestupem firmy po úspěchu procesoru Intel 8080, byl však vývoj programového vybavení u firmy v pozadí, a představa mikropočítače vybaveného operačním systémem příliš vzdálená. G. Kildall však pracoval na realizaci návrhu CP/M dál ve vlastní režii. Nezávislost na firmě Intel zřejmě prospěla přenosnosti systému, a tím jeho širokému rozšíření.

V téže době (přibližně kolem r. 1974) navrhl John Torode, pracující u firmy Digital Systems, řadič jednotky pružných disků, který umožnil perfektní spolupráci CP/M s diskovou pamětí. Předvádění prototypové verze CP/M mělo značný ohlas nejen mezi nadšenými vyznavači mikropočítačových systémů, ale zájem projevily i nově vznikající firmy, produkující mikropočítačové stavebnice a systémy. Tím vznikla potřeba instalovat operační systém CP/M na různých mikropočítačových systémech.

V letech 1974 až 1976 přepracoval G. Kildall systém CP/M tak, že z něj vyčlenil části závislé na konkrétním technickém prostředí. Systém se tak rozdělil na technicky závislou a technicky nezávislou část s přesně definovaným rozhraním. Při přenosu systému bylo nutno přizpůsobit technicky závislou část novému prostředí, zbytek systému zůstal beze změn, včetně programového prostředí a řídicího jazyka. Vznikla verze označovaná CP/M 1.3, která byla jako první zveřejněna a otevřela CP/M brány do světa uživatelů. V roce 1976 založil G. Kildall firmu Digital Research, která distribuuje CP/M a související programy.

Další vývoj CP/M byl ovlivněn především vývojem diskových pamětí. Vzniká verze CP/M 1.4, která dovoluje ovládat standardní diskety formátu IBM 3740 – osmipalcový, jednostranný pružný disk, s jednoduchou hustotou záznamu. Široké použití těchto disket dovolilo a stále dovoluje vysokou přenosnost souborů mezi jednotlivými instalacemi CP/M, a tím nebývalou přenosnost programových produktů.

Vývoj pružných disků se však nezastavil – vznikají minidiskety a mikrodiskety s rozmanitými formáty záznamu. Vznikají rovněž pevné disky s velkými kapacitami, použitelné v mikropočítačových systémech. Systém CP/M byl proto dále upraven tak, aby dovolil ovládat co nejširší třídu diskových pamětí – do technicky závislé části byly zabudovány tabulky popisující vlastnosti použitých disků. Příslušná verze nese označení CP/M 2.2, a je nejrozšířenější verzí systému CP/M.

S rostoucími nároky na schopnosti mikropoočítačového systému a klesající cenou pamětí vznikají systémy s vnitřní pamětí přesahující standardní adresní prostor 8bitových mikroprocesorů (tj. 64 KB, neboť v instrukcích je pro adresu vyhrazeno 16 bitů). Rozšíření operační paměti může být technicky provedeno např. tak, že paměť je rozdělena do stránek velikosti standardního adresního prostoru, které je možno dynamicky přepínat (banked memory). Verze CP/M 3.0 (též CP/M PLUS) podporuje ovládání stránkované paměti. Mimo to obsahuje řadu nových možností a vylepšení (systém nápověd, ochranná hesla apod).

Operační systém CP/M je (vzhledem k svému původu a poslání) monouživatelský a monoprogramový systém. Existují však varianty CP/M, které dovolují spouštění dalších úloh na pozadí (Concurrent CP/M), multiuživatelský přístup z více terminálů (MP/M), či sdílení prostředků v rámci lokální sítě (CP/NET). Pro 16bitové mikropočítače typu IBM PC jsou k dispozici verze Concurrent CP/M–86 a Concurrent DOS, které dovolují paralelní spuštění více úloh. Z důvodů kompatibility lze tyto verze doplnit o emulátor služeb operačního systému MS–DOS.

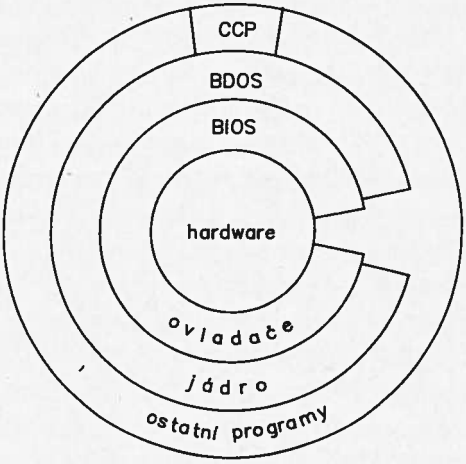
CP/M je velmi jednoduchý, ale přitom poměrně mocný operační systém, který lze snadno přizpůsobit a rozšiřovat. Bylo by jistě možné navrhnout systémy, které by byly z jistých hledisek rafinovanější než CP/M. Otázkou však je, zda by se podařilo zachovat pružnost při tak malých nárocích na paměť, jako má CP/M. Jedná se o velmi účelný a efektivní kompromis mezi mocností funkcí a minimalizací nároků na paměť, což je zejména u 8bitových mikropočítačů velmi důležité. Rovněž strategie využívání diskového média je jednoduchá, nevyžaduje programovou údržbu disků (komprese apod.) a přitom je velmi spolehlivá. Nezanedbatelným rysem CP/M je otevřenost systému vzhledem k uživateli, která jistě přispěla k jeho oblíbenosti. Celková struktura systému je navržena natolik chytře, že byla vzorem pro mnoho jiných operačních systémů pro mikropočítače.

3.2 Architektura systému CP/M

Jednou z charakteristických vlastností systému CP/M je hierarchicky vrstvená architektura. Programové vybavení je z hlediska systému rozděleno do tří vrstev, jak je naznačeno na obr. 3.1. Vrstvy bližší technickým

prostředkům jsou chápány jako hierarchicky nižší a obráceně. Uspořádání tedy vyjadřuje stupeň nezávislosti na konkrétních technických prostředcích. Nižší vrstvy poskytují služby vrstvám vyšším.

Nejnižší vrstva, která zajišťuje přímý styk s technickými prostředky počítače, se nazývá *základní systém vstupu a výstupu* – BIOS (Basic Input/Output System). Tato vrstva je závislá na technickém vybavení a je nutno ji pro každé technické prostředí speciálně upravit. Z hlediska terminologie operačních systémů představuje vrstva BIOS programový modul, který realizuje správu přídavných zařízení. Skládá se ze sady ovladačů (tzv. drivers), zajišťujících fyzický vstup a výstup dat.



Obr. 3.1. Struktura operačního systému CP/M

Jádro systému tvoří druhou vrstvu, nazývanou *základní diskový operační systém* – BDOS (Basic Disk Operating System), která je již nezávislá na konkrétním technickém prostředí. S technickými prostředky počítače spolupracuje prostřednictvím služeb vrstvy BIOS. V terminologii operačních systémů realizuje programový modul BDOS správu virtuálních prostředků, tj. především logický vstup a výstup, zejména pak systém ovládání souborů.

Veškeré ostatní programové vybavení tvoří z hlediska systému CP/M hierarchicky nejvyšší vrstvu, která s technickými prostředky komunikuje prostřednictvím nižších vrstev. Nejvyšší vrstva se skládá ze *samostatných programů*. Patří sem všechny aplikační a uživatelské programy, ale i sada předem připravených služebních programů, určených pro standardní činnosti na počítači a distribuovaných jako součást systému CP/M.

Do nejvyšší vrstvy patří i program, který zprostředkuje základní styk uživatele se systémem. Nazývá se *procesor příkazů* – CCP (Console Command Processor). Procesor příkazů čte a interpretuje příkazy zadávané systému uživatelem, tj. zpravidla realizuje požadavek na spuštění některého programu z nejvyšší vrstvy.

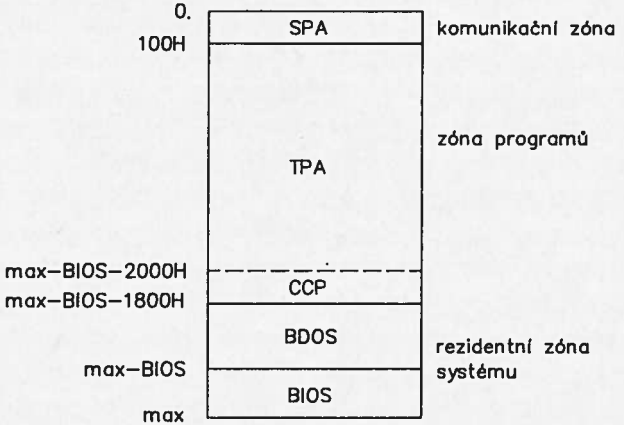
Všechny služby nižších složek architektury (tj. modulů BDOS a BIOS) jsou jednoduše přístupné programům. Vzhledem k hierarchické struktuře systému je však žádoucí, aby hierarchicky vyšší složky architektury využívaly pokud možno výhradně služby bezprostředně podřízené vrstvy. Programy nejvyšší vrstvy by proto měly používat výhradně služeb modulu BDOS a nekomunikovat přímo s modulem BIOS nebo dokonce s technickými prostředky. Jen tak lze totiž dosáhnout úplné přenosnosti programů, tj. jejich nezávislosti na konkrétní instalaci systému. Proto jako služby jádra systému CP/M prezentujeme pouze služby modulu BDOS; služby modulu BIOS považujeme za interní služby systému CP/M. V multiuživatelských verzích je modul BIOS nahrazen modulem XIOS (eXtended BIOS), kde přímé volání služeb modulu XIOS nelze doporučit.

Technické prostředky 8bitových mikropočítačů však obvykle neumožňují zabránit programům, aby komunikovaly přímo s modulem BIOS nebo s technickými prostředky. Na obr. 3.1 je tato skutečnost zvýrazněna výřezem v pravém okraji vrstev BDOS a BIOS. Programátor, který takový program vytváří, si však musí být vědom všech důsledků plynoucích z obcházení hierarchické struktury systému.

Poslední nezbytnou složkou systému CP/M, stojící mimo vrstvenou strukturu systému, je *systémový zavaděč* (značený CP/M Loader). Jeho úkolem je vytvořit v operační paměti počáteční obraz systému a předat mu řízení. Systémový zavaděč nemůže při své činnosti využívat služeb jádra, které v paměti teprve vytváří. Patří proto mezi technicky závislé složky systému, neboť musí komunikovat přímo s technickými prostředky počítače.

Důsledkem hierarchické architektury systému je skutečnost, že všechny programy pracující v prostředí operačního systému CP/M vyžadují ke své činnosti jádro systému (modul BDOS). Samo jádro pak využívá služeb ovladačů (modulu BIOS), které komunikují s technickými prostředky. Proto je jádro systému CP/M spolu s ovladači (tj. moduly BIOS a BDOS) umístěno v operační paměti rezidentně. Ostatní programy (včetně procesoru příkazů CCP) jsou uloženy na diskovém médiu a střídají se ve zbylé části paměti. Pro styk s prostředím využívají služeb rezidentního jádra.

Na obr. 3.2 je znázorněno rozdělení operační paměti v systému CP/M na zóny pevně přidělené jednotlivým složkám architektury. V rezidentní zóně systému na nejvyšších dostupných adresách jsou umístěny modul



Obr. 3.2. Rozdělení operační paměti v CP/M (SPA – System Parametr Area, TPA – Transient Program Area, CCP – Console Command Procesor, BDOS – Basic Disk Operating System, BIOS – Basic Input/Output System)

BDOS a technicky závislý modul BIOS, jehož velikost se v různých instalacích systému může měnit. Důvodem pro umístění rezidentní zóny na konec operační paměti je odstranění závislosti programů na aktuální velikosti rezidentní části systému. Zóna na počátku operační paměti slouží pro komunikaci programu se systémem přes pevné adresy.

Podrobný popis jednotlivých složek architektury je uveden v následujících kapitolách. Popisem procesoru příkazů (modulu CCP) se zabývá

kap. 4, která je zaměřena na uživatelské prostředí systému CP/M, včetně standardních služebních programů. Služby modulu BDOS jsou probírány v kap. 5, která je věnována programovému prostředí systému CP/M. Struktura a služby modulu BIOS jsou obsahem kap. 6, kde se též zabýváme možnostmi modifikace a instalace systému. Shrnutí informací potřebných při práci se systémem je uvedeno v přílohách.

3.3 Technické prostředí systému CP/M

Operační systém CP/M lze instalovat na poměrně rozmanité třídě mikropočítačů. Požadavky, kladené na technické prostředí pro instalaci systému CP/M, shrneme v tomto článku.

Systém CP/M je diskový operační systém, a proto se předpokládá, že domovem systému je *diskové médium*. Prvním nezbytným požadavkem na prostředí, v němž má systém CP/M pracovat, je existence alespoň jedné (třeba i virtuální) diskové jednotky, se kterou systém spolupracuje a z níž je zaváděn do operační paměti. Tato jednotka se nazývá *systémová disková jednotka* a bývá označena písmenem A (v rámci CP/M). Systém umí spolupracovat až se šestnácti (verze 1.x pouze se čtyřmi) diskovými jednotkami.

Druhým nezbytným požadavkem na prostředí je existence *technického zavaděče* (tzv. bootstrap), který umožní zahájení práce systému. Často používaným řešením je technický zavaděč umístěný ve stínové paměti ROM. Technický zavaděč obvykle přečte dohodnutý sektor (tzv. sektor zavaděče – BOOT sektor) z diskového média v systémové jednotce, uloží jeho obsah na dohodnutou adresu v operační paměti a předá mu řízení. Sektor zavaděče obsahuje systémový zavaděč (CP/M loader), který zavede vlastní systém CP/M do operační paměti z dohodnuté oblasti na disku a spustí. Z těchto důvodů je na systémovém diskovém médiu vyhrazen určitý prostor (obvykle stopy 0 a 1) jako domovská oblast systému, z níž je systém čten a ukládán do paměti. U datových disků tato oblast není použita (podrobněji se způsobem uložení informace na disku zabývá kap. 5).

Standardní varianta CP/M–80 je distribuována v kódu procesoru Intel 8080. Třetím nezbytným předpokladem je proto přítomnost *procesoru*, který umí *zpracovat kód Intel 8080* (tj. Intel 8080, Intel 8085 nebo Z80).

Přesněji, v kódu Intel 8080 jsou distribuovány moduly BDOS a CCP, moduly BIOS a systémový zavaděč (CP/M loader) je nutno přizpůsobit pro konkrétní technické prostředí a mohou proto využívat plný repertoár instrukcí daného procesoru. Podobně pro variantu CP/M–86 je nezbytný procesor typu Intel 8086, pro variantu CP/M–68K procesor typu Motorola 68000 apod.

Pro uložení kódu systému a provozovaných programů je nezbytná operační paměť RWM (RAM) o kapacitě minimálně 32 KB (pro verzi 1.4 postačí 20 KB). Doporučený rozsah paměti je samozřejmě 64 KB, výjimečně lze využívat paměť větší (CP/M PLUS, Concurrent CP/M). Standardní verze CP/M vyžaduje operační paměť RWM adresovatelnou od 0, neboť systém tento úsek paměti využívá pro komunikaci s programy. Některé speciální verze systému dovolují využívat i technické prostředí, kde je počátek adresního prostoru obsazen pamětí ROM. Příslušná úprava systému však vyžaduje úpravy ve zdrojových textech modulů BDOS a CCP, a není proto uživatelsky podporována (provádí ji sama firma Digital Research).

Posledním předpokladem pro instalaci systému CP/M je přítomnost zařízení, které umožňuje inteligentní komunikaci s uživatelem – tzv. *systémové konzole* v terminologii systému CP/M. Obvykle se jedná o zobrazovací jednotku s klávesnicí, často připojenou jako terminál mikropočítače. Protože však termín terminál má svou vlastní interpretaci, budeme důsledně používat termínu systémová konzole. Mimo systémovou konzoli může systém CP/M spolupracovat s řadou dalších zařízení (maximum je 15 zařízení, samozřejmě kromě diskových jednotek).

Technicky nezávislé složky systému CP/M (moduly BDOS, CCP a služební programy) není nutno před instalací systému upravovat. Na druhé straně je provozování systému CP/M v daném technickém prostředí podmíněno přizpůsobením modulu BIOS a systémového zavaděče. Přizpůsobením těchto složek systému CP/M technickým podmínkám se zabývá kap. 6.

Operační systém CP/M–80 je koncipován tak, aby nevyžadoval od technických prostředků počítače realizaci přerušovacího systému. Lze jej proto instalovat i na počítačích, které přerušovací systém zabudován nemají. Technicky nezávislé složky systému přerušení nevyužívají. Systém se překrývá pouze s položkou pro přerušení 0, kde je nahrána instrukce skoku na sekvenci pro startování systému. Technicky závislé složky systému mohou využívat všechna ostatní přerušení. Jistou výjimku představuje položka pro

přerušení 7 (adresa 38H), kterou využívá služební program DDT přes instrukci RST 7 a jejímu používání je tedy lépe se vyvarovat.

Splňuje-li technické prostředí nezbytné předpoklady a jsou-li k dispozici přizpůsobené verze modulů BIOS a systémového zavaděče, můžeme vložit médium se systémem CP/M do dohodnuté systémové jednotky a uvést v činnost technický zavaděč.

3.4 Zavádění a startování systému

Systém CP/M rozeznává dva způsoby startování. První způsob je nazýván *studený start* systému (cold boot) a spočívá v úplné inicializaci technického prostředí i operačního systému. Používá se především při prvním odstartování systému po zapnutí počítače (odtud název studený start). Studený start systému je rovněž nutno použít po vážných chybách, způsobených technickými nebo programovými prostředky (např. přepsáním kódu modulu BIOS).

Druhý způsob startování systému CP/M je nazýván *teplý start* (warm boot), neboť se jedná o obnovení činnosti systému při běžícím počítači, zajišťované výhradně programovými prostředky modulu BIOS. Teplý start je proto možný pouze tehdy, není-li porušen kód modulu BIOS.

Studený start systému je nutno vyvolat technickými prostředky – zapnutím počítače, stiskem tlačítka INIT, RESET apod. Technické prostředky aktivují technický zavaděč (bootstrap), který zavede do paměti systémový zavaděč (CP/M loader) a předá mu řízení. Systémový zavaděč nahraje do operační paměti kód modulů CCP, BDOS a BIOS ze systémové oblasti média v systémové jednotce. Poté vyvolá interní službu modulu BIOS nazývanou studený start (BOOT). Tato služba je vyvolána během činnosti systému vždy právě jen jednou při studeném startu systému. Úkolem této služby je především příprava technického prostředí, tj. naprogramování (nastavení) stykových obvodů apod. Současně jsou inicializovány všechny vnitřní proměnné systému a je nastaven obsah komunikační zóny SPA. Služba studeného startu obvykle zobrazí na systémové konzoli úvodní zprávu. Doporučený tvar úvodní zprávy je:

xxK CP/M VER m.n

kde: xx je velikost paměti, kterou tento systém ovládá (16 až 64 KB),
m.n je verze systému (implicitně předpokládáme verzi 2.2, pokud není uvedeno jinak).

Na závěr studeného startu je aktivován procesor příkazů CCP. Aktivace CCP se projeví zobrazením výzvy k zadání příkazu pro systém ve tvaru:

```
A>
```

Písmeno A ve výzvě indikuje tzv. *aktuální diskovou jednotku*. Při studeném startu systému je jako aktuální nastavena jednotka, z níž byl systém zaveden, tj. systémová disková jednotka. Dále již uživatel komunikuje se systémem pomocí řídicího jazyka (viz popis řídicího jazyka v kap. 4).

Teplý start systému je aktivován programově vyvoláním služby modulu BIOS nazývané teplý start (WBOOT). Akceptuje-li systém vstup ze systémové konzole, lze teplý start systému vyvolat uživatelsky stiskem kombinace kláves CTRL–C (zobrazí se jako ^C). Kombinace CTRL–C musí být stisknuta jako první v řádku, jinak ji systém neakceptuje. Tento postup lze např. doporučit při výměně média v diskové jednotce.

Při teplém startu je do operační paměti nahrán znovu pouze kód modulů CCP a BDOS. Po nezbytné inicializaci systémových proměnných a nastavení obsahu komunikační zóny SPA je reaktivován monitor CCP. Na rozdíl od studeného startu nemění se při teplém startu aktuální disková jednotka, nastavená v komunikační zóně SPA. Výzva monitoru po teplém startu proto může obsahovat označení jiné jednotky než A.

Poznamenejme, že procesor příkazů CCP obsahuje interně dva vstupní body, aby bylo možno rozeznat studený a teplý start. Této skutečnosti lze využít pro automatické odstartování programu, který má být proveden právě jednou při studeném startu. Realizace bude probrána v kap. 6.

3.5 Uživatelské prostředky systému CP/M

Uživatel operačního systému CP/M má k dispozici řídicí jazyk systému, služební programy a svou sadu aplikačních uživatelských programů. Řídicí jazyk systému definuje a interpretuje procesor příkazů CCP. S jeho pomocí může uživatel spustit jednu úlohu (provést jeden program), nebo vykonat připravenou dávku úloh. Služební programy provádějí omezenou

sadu základních úloh, jako zobrazení a nastavení stavu systému, kopírování informace a přípravu programů v asembleru.

Operační systém CP/M dále poskytuje uživateli možnost pracovat s jistou sadou *logických zařízení*. Tato sada je rozdělena na dvě kategorie – znaková zařízení a bloková zařízení. *Znaková zařízení* slouží pro sekvenční vstup a výstup informace po jednotlivých znacích (slabikách). Pro *bloková zařízení* je charakteristický přenos informace po větších blocích pevné délky a přímý přístup k blokům podle čísla bloku (adresy). Obecně jsou tedy bloková zařízení chápána jako vnější paměti s přímým přístupem. V terminologii systému CP/M jsou bloková zařízení nazývána *jednotky* (drives).

Systém CP/M zajišťuje pro bloková zařízení možnost vytváření a údržby logických informačních celků nazývaných soubory. *Soubory* jsou identifikovány jménem a typem, zařízení jsou identifikována znakovým řetězcem ukončeným znakem " : ".

3.5.1 Logická znaková zařízení

V rámci CP/M jsou definována čtyři logická znaková zařízení, označená zkratkami:

CON: Systémová konzole pro vstup a výstup (CONsole),
RDR: Logické zařízení znakového vstupu (ReaDeR),
PUN: Logické zařízení znakového výstupu (PUNcher),
LST: Logické zařízení pro tisk (LiST).

Označení logických znakových zařízení se traduje z původních verzí systému. Těmto logickým zařízením mohou být přiřazena různá fyzická zařízení. Povolená označení fyzických zařízení jsou:

TTY: pomalá konzole typu dálnopis (TeleTYpe),
CRT: rychlá konzole – obvykle obrazovka s klávesnicí (Cathod Ray Tube),
BAT: pseudozařízení, které lze využít při zpracování v dávkách (BATch); znamená, že vstup ze zařízení CON: je přesměrován na zařízení RDR: a výstup na zařízení CON: je přesměrován na zařízení LST:,
UC1: uživatelsky definovaná konzole (User defined Console 1),

PTR: snímač děrné pásky (Paper Tape Reader),
UR1: uživatelsky definované vstupní zařízení č.1 (User defined Reader 1),
UR2: uživatelsky definované vstupní zařízení č.2 (User defined Reader 2),
PTP: děrovač děrné pásky (Paper Tape Punch),
UP1: uživatelsky definované výstupní zařízení č.1 (User defined Punch1),
UP2: uživatelsky definované výstupní zařízení č.2 (User defined Punch 2),
LPT: tiskárna (Line PrinTer),
UL1: uživatelsky definovaná tiskárna (User List device 1).

Aktuální přiřazení fyzických zařízení logickým je možné zobrazit a nastavit služebním programem STAT, nebo programově pomocí služeb jádra systému. Danému logickému zařízení lze přiřadit jen určitá fyzická zařízení podle tab. 3.1.

Tab. 3.1. Možná přiřazení logických a fyzických zařízení

Logické zařízení	Fyzické zařízení			
CON:	TTY:	CRT:	BAT:	UC1:
RDR:	TTY:	PTR:	UR1:	UR1:
PUN:	TTY:	PTP:	UP1:	UP2:
LST:	TTY:	CRT:	LPT:	UL1:

Např. logickému zařízení LST: můžeme přiřadit pouze fyzická zařízení TTY:, CRT:, LPT: nebo UL1:, neboť ta umožňují výstup informace. Aktuální přiřazení logických a fyzických zařízení je uloženo ve stavové slabice nazývané IOBYTE (pouze pro verze 2.x a nižší), jejíž stav můžeme číst a měnit pomocí služeb modulu BDOS (viz kap. 5). Rovněž spolupráce s logickými zařízeními je možná pomocí služeb modulu BDOS, který je převádí na volání služeb modulu BIOS.

Při instalaci systému CP/M je definováno, jaká zařízení a jakým způsobem modul BIOS ovládá (viz kap. 6). Z uvedeného je zřejmé, že v dané konkrétní konfiguraci může modul BIOS ovládat jedno až patnáct konkrétních zařízení. Minimálně musí modul BIOS obsahovat ovladač

fyzického zařízení TTY: a všechna ostatní fyzická zařízení programově simulovat pomocí zařízení TTY:. Maximálně může modul BIOS obsahovat patnáct ovladačů, z nichž jsou ovšem v každém okamžiku přístupné vždy právě čtyři, přiřazené v tomto okamžiku čtyřem logickým zařízením. Změnou přiřazení lze využívat různé čtveřice ovladačů, přičemž např. zařízení TTY: přiřazené logickému zařízení CON: může zastupovat jiný ovladač, než zařízení TTY: přiřazené logickému zařízení LST:.

3.5.2 Diskové jednotky a soubory

Systém CP/M (resp. jeho modul BDOS) zajišťuje logickou organizaci dat na diskových jednotkách (obecněji na vnějších pamětech s přímým přístupem). Z hlediska uživatele je k dispozici maximálně 16 diskových jednotek, označených v rámci systému písmeny A až P. Protože každá identifikace zařízení je ukončena znakem " : ", jsou diskové jednotky identifikovány označením A: až P:.

Systém rozeznává dvě speciální diskové jednotky, systémovou a aktuální. *Systémová jednotka* je určena pevně a systém předpokládá, že na médiu v systémové jednotce je uložen kód systému. Ze systémové jednotky je kód systému čten do paměti při startování systému. Obvykle bývá jako systémová použita jednotka A, ale úpravou modulu BIOS lze systémovou jednotku změnit. *Aktuální jednotkou* může být libovolná jednotka (tedy i systémová) a lze ji měnit pomocí uživatelských příkazů. Zavedení aktuální jednotky slouží pro zkrácení značení, aby při odkazu na diskovou jednotku nebylo nutno vždy označení jednotky uvádět. Aktuální jednotka je použita implicitně vždy, pokud není explicitně požadována jednotka jiná. Označení aktuální jednotky je pro přehled uvedeno ve výzvě systému.

Systém CP/M dále zavádí termín *aktivní jednotka.* Systém označí jednotku jako aktivní tehdy, pokud s ní již spolupracoval a má o médiu v ní vloženém zaznamenány jisté informace v operační paměti. Pro aktivní jednotku zná systém např. kontrolní součet jistého úseku adresáře média a před každým zápisem kontroluje, zda odpovídající úsek adresáře na médiu má shodný kontrolní součet. Pokud ne, předpokládá se, že došlo k nepovolené výměně média. Z toho vyplývá, že regulární výměnu média je možno

provést pouze v jednotce, která není aktivní. Deaktivací lze provést uživatelsky teplým startem systému (CTRL–C) pro všechny jednotky, nebo programově pomocí služeb jádra i pro jednotlivé jednotky.

Médium, umístěné v diskové jednotce, je dále děleno na *logické soubory*. Logický soubor v systému CP/M se skládá z logických vět pevné délky 128 slabik, ke kterým je možno přistupovat sekvenčně nebo přímo na základě čísla věty. Věty se číslují sekvenčně počínaje 0. Soubory je možno vytvářet, rušit, vyhledávat, zapisovat do nich nebo z nich číst po logických větách. Maximální teoretická velikost jednoho souboru je 65 536 vět (8 MB). Čísla vět se proto pohybují v rozsahu 0 až 65 535.

Stejné médium může využívat více uživatelů. Systém CP/M rozeznává až 16 uživatelů identifikovaných čísly 0 až 15. Z nich vždy právě jeden uživatel je tzv. *aktuální uživatel*. Nastavení a zjištění aktuálního uživatele lze zajistit pomocí systémových příkazů (viz popis příkazů USER, STAT) nebo programově pomocí služeb jádra. V některých verzích systému (MP/M, Concurrent CP/M–86) je identifikace aktuálního uživatele uvedena i ve výzvě systému. Pokud je např. aktuální uživatel číslo 0 a jako aktuální je nastavena jednotka A, má systémová výzva tvar A> (příp. 0A>). Je-li pro stejnou jednotku nastaven jako aktuální uživatel číslo 3, má výzva tvar 3A>.

Každé diskové médium je proto možno rozdělit na 16 logických oblastí pro jednotlivé uživatele. Systém vždy pracuje pouze s tou oblastí na médiu, která přísluší právě aktuálnímu uživateli. Obsah oblastí jiných uživatelů není standardním způsobem přístupný (výjimky je možno nalézt v popisu příkazu PIP). Rozdělení média na oblasti zajišťuje automaticky systém. Aktuální stav média je možno zjistit pomocí systémových příkazů (viz popis STAT).

3.5.3 Identifikace souborů

Soubor vložený na vnější paměti je identifikován trojicí:

[jednotka:] jméno [.typ] ,

kde: jednotka je označení diskové jednotky (písmena A až P u verze 2.x a verzí vyšších, A až D u verze 1.4 a verzí nižších),
jméno je jméno souboru (maximálně 8 znaků),
typ je typ souboru (maximálně 3 znaky).

U verze 3.0 a vyšších (CP/M PLUS, Concurrent CP/M atd.) lze dále ke každému souboru přiřadit heslo chránící přístup k souboru; heslo je maximálně osmiznakové a uvádí se na konci označení souboru oddělené znakem " ; ". Úplný tvar identifikace souboru má pak formu:

[jednotka:] jméno [.typ] [;heslo]

Předpona jednotka: může být vynechána – v tom případě se uvažuje aktuální disková jednotka (uvedená ve výzvě systému, např. A>). Přípona .typ není rovněž povinná – obvykle se však používá k rozlišení obsahu uložené informace.

3.5.4 Konvence pro značení typů souborů

Systém rozeznává tzv. *povinné typy souborů*, které sám využívá a není možno je v rámci systému obejít. Jsou to zejména typy COM a SUB. Každý program zaváděný a spouštěný monitorem CCP musí být povinně typu COM a program uložený v souboru jiného typu nelze monitorem spustit. Dávky příkazů jsou povinně uloženy v souborech typu SUB a jiný typ souboru služební program SUBMIT jako dávku nepřijme.

Mimo tyto základní povinné typy existují další povinné typy, používané ve služebních programech. Např. standardní asembler ASM předpokládá zdrojový text pouze v souborech typu ASM.

Mimo povinné typy souborů existuje řada konvencí pro používání typů. Rezervované a běžně užívané typy jsou např. (může se lišit dle produkující firmy):

ASM – zdrojový program pro asembler ASM,
BAK – záložní kopie souboru (např. při editaci),
BAS – zdrojový program v jazyce Basic,
CBL – zdrojový program v jazyce Cobol,
CMD – zaveditelný program pro procesor Intel 8086 nebo procedura pro databázi dBASE,
COM – absolutní zaveditelný program,
DAT – datový soubor,
DBF – datový soubor pro databázi dBASE,
DOC – soubor obsahující dokumentaci,

FOR – zdrojový program v jazyce Fortran,
HEX – hexadecimální formát (viz přílohy – výstup ASM zaveditelný pomocí LOAD),
LIB – knihovna nebo vsuvka,
LST(PRN) – soubor určený k tisku (protokol o překladu),
MAC – zdrojový program v jazyce Macro–80,
NDX – indexový soubor databáze dBASE,
OVR – soubor obsahující překryvné segmenty,
PAS(SRC) – zdrojový program v jazyce Pascal (Pascal/MT+),
PLI – zdrojový program v jazyce PL/I,
PRG – procedura pro databázi dBASE ve verzi CP/M–86,
REL(ERL) – modul v jazyce relativních adres,
SAV – systémový soubor (verze 2.x),
SUB – dávka příkazů (viz SUBMIT),
SYM – tabulka symbolů,
TXT(TEX) – textový soubor,
XRF – křížové reference,
$$$ – pracovní soubor.

V identifikaci souboru se mohou vyskytovat pouze písmena, číslice nebo znaky:

+ - ! @ $ % ^ & ( ) _ ~ ` | { } " '

Uvádíme-li identifikaci souboru v rámci příkazů pro monitor CCP, monitor automaticky převádí malá písmena na velká. Ostatní znaky mají speciální význam v rámci příkazů CP/M. Následující znaky nejsou proto v identifikaci souboru povoleny:

< > . , ; : = ? * [ ]

Poznámka: Pokud vytvoříme nějakým programem soubor obsahující v identifikaci malá písmena (např. Muj.TXT), není tento soubor přístupný přes příkazy prostřednictvím monitoru – nelze jej např. zrušit příkazem ERA s udáním jednoznačného jména Muj.TXT (zruší se příkazem ERA * . TXT, ale spolu s ostatními soubory typu TXT).

3.5.5 Identifikace skupin souborů

V rámci CP/M lze identifikovat nejen určitý jediný soubor, ale i celou skupinu souborů – např. chceme zrušit všechny záložní kopie souborů, tj. skupinu souborů typu .BAK. Pro tento účel jsou vyhrazeny znaky " ? " a " * " (wild cards), s jejichž pomocí lze vytvářet identifikaci skupiny souborů. Identifikace obsahující znaky " ? " a " * " označuje skupinu souborů, do níž patří všechny soubory, jejichž identifikace se liší pouze v místě těchto speciálních znaků. Na místě znaku " ? " může stát libovolný znak, na místě znaku " * " libovolná posloupnost znaků. Znaky " ? " a " * " nelze použít v označení diskové jednotky ani v hesle. Identifikaci souboru, která neobsahuje znaky " ? " nebo " * ", budeme nazývat jednoznačná identifikace souboru a označovat ufn (unambiguous file name). Identifikaci skupiny souborů budeme značit afn (ambiguous file name).

Součástí úplné identifikace souboru (pro verze 2.x a vyšší) je i určení uživatele, do jehož oblasti soubor přísluší. V intencích za vedení uživatelských oblastí však je znemožnění přístupu k souborům jiných uživatelů. Proto určení uživatele není explicitně možno v identifikaci souboru ani skupiny souborů uvést. Každá identifikace se vždy vztahuje k oblasti implicitně stanoveného aktuálního uživatele.

Příklady

Jednoznačná jména souborů (ufn):

ALFA	soubor ALFA na aktuální diskové jednotce v oblasti aktuálního uživatele
ALFA.SRC	dtto ALFA.SRC
B:ALFA.LST	soubor ALFA.LST na diskové jednotce B v oblasti aktuálního uživatele
A:ALFA1.LS	jiný soubor na jiném disku, ale v oblasti téhož uživatele

Označení skupiny souborů (afn):

.	všechny soubory na aktuální diskové jednotce v oblasti aktuálního uživatele
*.BAK	všechny soubory typu BAK na aktuální diskové jednotce v oblasti aktuálního uživatele
B:ALFA.*	všechny typy souboru ALFA na diskové jednotce B v oblasti aktuálního uživatele
A*.*	všechny soubory se jménem začínajícím na A na aktuální jednotce v oblasti aktuálního uživatele
A???????.???	totéž
A??.COM	všechny soubory typu COM se jménem začínajícím znakem A a ne delším než tři znaky, které se nacházejí na aktuální diskové jednotce v oblasti aktuálního uživatele.

4. Uživatelské prostředí systému CP/M

Uživatelské prostředí systému CP/M vytvářejí především procesor příkazů CCP a služební programy. Procesor příkazů CCP definuje řídicí jazyk systému, zajišťuje komunikaci s uživatelem a interpretuje základní sadu příkazů. Služební programy doplňují sadu příkazů o prostředky pro standardní akce a dotvářejí tak uživatelské prostředí systému. Uživatel systému CP/M obvykle nepracuje pouze se standardními systémovými prostředky, ale využívá i jistou sadu aplikačně orientovaných programů. Každý takový aplikační program navržený pro systém CP/M si při komunikaci s uživatelem vytváří své vlastní prostředí. Protože zde nemůžeme popsat všechny možné aplikační programy, věnujeme se v této kapitole výhradně popisu standardních systémových prostředků uživatele operačního systému CP/M.

4.1 Řídicí jazyk systému CP/M

Řídicí jazyk operačního systému CP/M je poměrně jednoduchý, což odpovídá monouživatelskému a monoprogramovému charakteru systému. Vzhledem k rozmanitému repertoáru zobrazovacích jednotek 8bitových mikropočítačů je řídicí jazyk orientován textově a nevyužívá speciálních grafických prostředků pro komunikaci.

Základní konstrukcí řídicího jazyka je příkaz pro spuštění úlohy. Příkazy se zadávají sekvenčně na základě výzvy systému. Navíc je možno sdružit posloupnost příkazů do tzv. *příkazové procedury* (v terminologii systému CP/M bývá označována jako *dávka příkazů*), kterou provádí systém automaticky. Jedná se o jistou formu makropříkazu na úrovni řídicího jazyka, včetně symbolických parametrů.

Tvar řídicích příkazů a struktura příkazových procedur je definována procesorem příkazů CCP. Kdykoli je aktivován procesor příkazů CCP (po odstartování systému, ukončení programu) je na systémové konzoli zobrazena výzva k zadání příkazu ve tvaru

X> .

Písmeno X ve výzvě indikuje aktuální diskovou jednotku. Při studeném startu je jako aktuální nastavena systémová jednotka A, při teplém startu (ukončení programu) se aktuální jednotka nemění. Aktuální jednotka funguje jako implicitní v těch případech, kdy neuvedeme v požadavcích na přístup k disku jednotku jinou.

4.1.1 Tvar příkazů CP/M

Obecná struktura příkazu pro CP/M vychází z představy, že zadáním příkazu požaduje uživatel provedení jisté úlohy. V příkazu proto musí být uvedena identifikace místa, kde je uložen prováděcí kód úlohy (program) a případné parametry, které modifikují činnost úlohy. Každý příkaz má obecně tvar:

[jednotka:] [jméno] [_ parametry] ,

kde dvojice jednotka a jméno identifikují soubor na diskovém médiu, obsahující prováděcí program. Není-li jednotka v příkazu uvedena, uvažuje se aktuální disková jednotka. Není-li uvedeno v příkazu ani jméno, jedná se o prázdný příkaz, který způsobí pouze opětovné zobrazení výzvy. Pokud jsou v příkazu uvedeny parametry, musí být odděleny právě jednou mezerou. Podrobný popis syntaxe příkazů CP/M je uveden v odst. 4.1.5.

Tvar parametrů uvedených v příkazu závisí na programu, který bude prováděn a který je zpracuje. Standardní doporučený tvar je posloupnost řetězců, vzájemně oddělených znakem " _ " (mezera). Procesor příkazů předá parametry programu v komunikační zóně.

Příkazy zadává uživatel ze systémové konzole po zobrazení výzvy systému nebo jsou čteny z připravené dávky příkazů, pokud se provádí příkazová procedura. Čtení příkazu ze systémové konzole zajišťuje procesor příkazů pomocí programové služby modulu BDOS (služba 10), která ukončí čtení a předá příkaz ke zpracování po stisku klávesy s kódem 13 (znak CR ve stan-

dardním kódu ASCII) nebo klávesy s kódem 10 (LF). Odpovídající klávesa je obvykle označena RETURN, CR, ENTER, CTRL–M, CTRL–J apod. dle typu klávesnice. Před odesláním příkazu je možno příkazový řádek editovat.

4.1.2 Editace systémových příkazů

Při zadávání příkazu se můžeme dopustit chyby. Z toho důvodu lze příkazový řádek před odesláním ke zpracování klávesou CR nebo LF editovat pomocí tzv. *editačních kláves.* Aby nebyly editační klávesy závislé na konkrétní klávesnici, využívá se zejména tzv. řídicích znaků, které klávesnice generuje při současném stisku klávesy CTRL a další klávesy. V zobrazení se řídicí znaky uvádějí předznačeny " ^ " ; např. CTRL–X se zobrazí jako ^X.

Klávesa	Funkce
RUBOUT/DEL	Zruší poslední platný znak v řádku, ale v zobrazení na systémové konzoli zrušený znak nemaže, naopak jej pro kontrolu zobrazí znovu (vhodné pro případ, kdy je jako systémová konzole použit např. dálnopis).
^H (BS)	Zruší poslední platný znak v řádku a v zobrazení na systémové konzoli zrušený znak vymaže zasláním posloupnosti znaků "návrat o pozici zpět", "mezera", "návrat o pozici zpět" (08H, 20H, 08H). (Implementováno ve verzi 2.x a vyšších; vhodné pro systémovou konzoli s obrazovkou.)
^X	Zruší doposud zadaný řádek a na systémové konzoli řádek vymaže (stejným postupem jako v případě ^H). Kurzor se vrátí do výchozí pozice a řádek je možno znovu zadat. (Vhodné pro systémovou konzoli s obrazovkou.)
^U	Zruší doposud zadaný řádek; na systémové konzoli řádek nemaže, ale ukončí jej znakem "#" a přejde na následující řádek pod první znak zrušeného řádku. (Vhodné pro systémovou konzoli typu dálnopis).

^R Přezobrazí celý vstupní řádek – ukončí aktuální zápis řádku znakem "#" , přejde na nový řádek pod první znak přezobrazovaného řádku a znovu řádek zobrazí (bez zrušených znaků – vhodné pro systémovou konzoli typu dálnopis, kde nelze mazat zrušené znaky).

^E Kurzor přejde na začátek dalšího řádku, ale příkazový řádek není odeslán – slouží k zadávání delších příkazů (maximální délka příkazu je 255 znaků).

^M (CR, ^J, LF, RETURN, ENTER, RET atd.) Ukončí vstup řádku a předá jej ke zpracování.

Poznámka: Editaci neprovádí procesor příkazů, ale zajišťuje ji přímo služba modulu BDOS (služba 10), jejímž prostřednictvím procesor CCP příkazovou řádku čte. Stejná pravidla platí i pro ostatní programy, které tuto službu modulu BDOS využívají, což lze doporučit jako vhodnou konvenci.

4.1.3 Zpracování příkazů

Způsob zpracování přečteného příkazu je odvozen z rozdělení příkazů CP/M na dvě základní skupiny:
– příkazy, které jsou často používány, jsou v jistém smyslu základní a nekladou velké požadavky na zpracování,
– ostatní příkazy.

Systém je řešen tak, že prováděcí kód příkazů první skupiny je přímo zabudován do procesoru příkazů CCP a příkazy této skupiny se proto nazývají *zabudované příkazy*. Všechny ostatní příkazy jsou zpracovány pomocí samostatných programů (uložených zcela běžným způsobem v datové oblasti disku). Tyto příkazy se nazývají *transientní příkazy*, neboť jejich kód je v paměti uložen přechodně po dobu provádění. Z tohoto hlediska lze každý další vytvořený program (spolupracující s prostředím přes služby CP/M) považovat za rozšíření repertoáru transientních příkazů CP/M.

Přečtený příkazový řádek procesor příkazů CCP analyzuje tak, že oddělí identifikační část příkazu, která je ukončena první mezerou nebo

koncem řádku. Pokud je příkazový řádek prázdný, zobrazí procesor příkazů výzvu k zadání dalšího příkazu.

Pro úspěšné analýze identifikační části porovná nejprve procesor příkazů jméno uvedené v identifikaci s vestavěnou tabulkou zabudovaných příkazů. Nalezne-li jméno v této tabulce, přejde na vykonání odpovídajícího zabudovaného příkazu. Případné označení jednotky, uvedené v příkazu, nemá na provedení zabudovaného příkazu vliv.

Pokud se nejedná o zabudovaný příkaz, pokouší se procesor příkazů nalézt prováděcí program. Nalezne-li v příkazu správné označení jednotky, pokusí se jednotku aktivovat. Není-li v příkazu jednotka uvedena, jedná se o aktuální jednotku, která je vždy aktivní. Po úspěšné aktivaci hledá procesor příkazů v adresáři soubor s identifikací jméno.COM (označení COM je zkratkou COMmands, tj. soubor obsahující instrukce). Typ COM je povinný a pomocí procesoru příkazů nelze zavést a spustit program, který není uložen v souboru typu COM. Přípona COM se v příkazu neuvádí.

V případě, že je soubor s identifikací jméno.COM nalezen, zkopíruje procesor příkazů CCP parametry uvedené v příkazu do komunikační zóny (podrobně je způsob uložení parametrů v komunikační zóně uveden (v kap. 6). Dále se procesor příkazů pokusí parametry interpretovat jako identifikaci souboru nebo dvě identifikace souborů oddělené mezerou. V úspěšném případě připraví v komunikační zóně ještě řídicí bloky pro tyto soubory. Poté procesor příkazů nahraje obsah souboru jméno.COM do operační paměti do zóny TPA (od adresy 100H) a předá mu řízení (instrukcí CALL 100H).

Není-li soubor jméno.COM nalezen, nebo byla-li v příkazu nalezena chyba (chybné označení jednotky, chybně zapsané jméno, parametry neoddělené mezerou apod.), hlásí procesor příkazů CCP chybovou zprávu ve tvaru

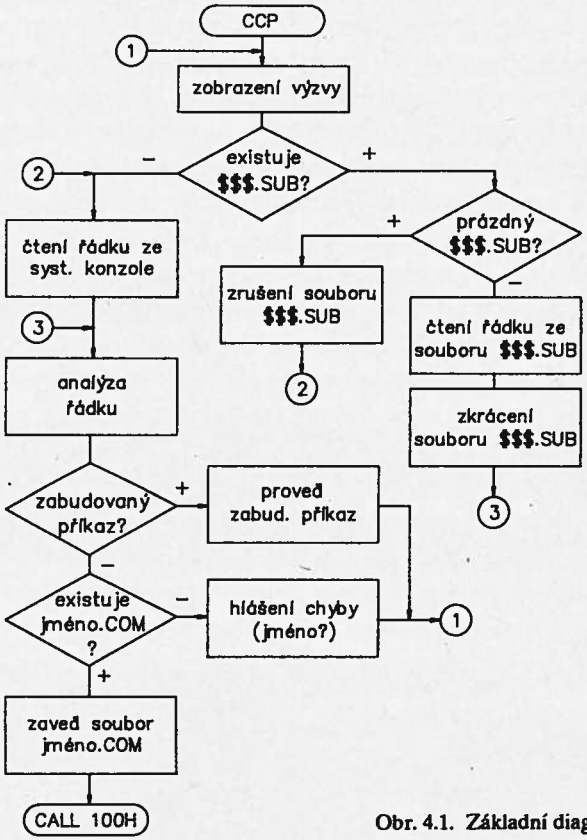
jméno?

a vyzve uživatele k zadání dalšího příkazu.

Po ukončení činnosti zavedeného programu je obvykle znovu aktivován procesor příkazů CCP (tj. obvykle znovu zaveden do paměti) a rovněž vyžaduje zadání dalšího příkazu.

4.1.4 Zpracování příkazových procedur

Procesor příkazů CCP dovoluje mimo zadávání příkazů ze systémové konzole též zpracování předem připravených příkazových procedur. Základní diagram činnosti procesoru je znázorněn na obr. 4.1. Po zobrazení výzvy hledá procesor příkazů CCP nejprve na systémové jednotce (A) soubor $$$.SUB. Pokud jej nalezne, přečte z něj poslední větu a interpretuje ji jako příkaz pro systém. Současně soubor A:$$$.SUB zkrátí o přečtenou větu. Pokud soubor A:$$$.SUB neexistuje (nebo byl právě zrušen), čte procesor příkaz ze systémové konzole, a čeká tedy na zadání příkazu od uživatele.



Obr. 4.1. Základní diagram činnosti procesoru příkazů CCP

Spolupráce se souborem A:$$$.SUB slouží pro zpracování příkazových procedur. Příkazová procedura je tvořena textem, jehož jednotlivé řádky představují příkazy pro systém CP/M. Pořadím řádků je stanovena požadovaná posloupnost provádění příkazů. Příkazové procedury jsou povinně uloženy v souborech typu SUB. Zpracování příkazové procedury se zadá příkazem SUBMIT ve tvaru

A>SUBMIT jméno [_ parametry] ,

kde jméno je označení příkazové procedury (uložené v souboru jméno.SUB) Při zadání zpracování příkazové procedury (viz popis příkazu SUBMIT), je na základě textu dávky vytvořen soubor $$$.SUB. Kvůli jednoduchému zpracování jsou řádky z textu dávky uloženy pozpátku a každý řádek je uložen do jedné logické věty souboru $$$.SUB. Při zpracování dávky jsou jednotlivé příkazy odebírány z konce souboru $$$.SUB až do jeho vyprázdnění nebo nalezení prázdného řádku (ten slouží jako příznak konce dávky).

Soubor $$$.SUB je vytvořen na aktuální jednotce. Pokud je aktuální jednotka současně systémovou jednotkou, je bezprostředně po ukončení zadání dávky aktivován procesor příkazů a jeho vstup přesměrován na soubor A:$$$.SUB. Pokud je aktuální jiná jednotka než systémová, je pouze vytvořen soubor $$$.SUB, ale k jeho zpracování dojde až tehdy, když je disk s tímto souborem vložen do systémové jednotky.

Během zobrazení výzvy na systémové konzoli zjišťuje procesor příkazů, zda byla stisknuta klávesa DEL (RUBOUT atd.). V kladném případě zruší procesor příkazů soubor A:$$$.SUB, čímž lze provádění dávky příkazů přerušit a ukončit.

4.1.5 Úplná syntaxe příkàzů CP/M

V tomto odstavci je uvedena úplná syntaxe příkazů CP/M v běžné BNF notaci. Syntaktické kategorie jsou vyznačeny pomocí úhlových závorek, např. <příkaz>. Pravidla BNF jsou zapisována ve tvaru

```
<kategorie> : : = varianta 1
              | varianta 2
              ...
              | varianta N ,
```

kde na levé straně je označení definované kategorie a na pravé straně jsou jednotlivé možné varianty. Znaky mimo úhlové závorky jsou povinné (včetně mezer).

```
<příkaz CP/M> : : =
        <identifikace diskové jednotky> <příkaz> <konec>
<příkaz> : : = <zabudovaný příkaz>
        | <transientní příkaz>
<zabudovaný příkaz> : : = <prázdný příkaz>
        | <příkaz DIR>
        | <příkaz ERA>
        | <příkaz REN>
        | <příkaz SAVE>
        | <příkaz TYPE>
        | <příkaz USER>
<prázdný příkaz> : : = <prázdný řetěz>
<příkaz DIR> : : =
        DIR
        | DIR<identifikace souboru>
        | DIR<identifikace skupiny souborů>
<příkaz ERA> : : =
        ERA<identifikace souboru>
        | ERA<identifikace skupiny souborů>
<příkaz REN> : : =
        REN<identifikace souboru> = <identifikace souboru>
<příkaz SAVE> : : = SAVE<počet> <identifikace souboru>
<příkaz TYPE> : : = TYPE<identifikace souboru>
<příkaz USER> : : = USER<číslo uživatele>
<transientní příkaz> : : =
        <příkaz ASM>
        | <příkaz DDT>
        | <příkaz DUMP>
        | <příkaz ED>
```

```
                | <příkaz LOAD>
                | <příkaz MOVCPM>
                | <příkaz PIP>
                | <příkaz STAT>
                | <příkaz SUBMIT>
                | <příkaz SYSGEN>
                | <jiný příkaz>

<příkaz ASM> : : =
                ASM<jméno souboru> . <parametry ASM>

<příkaz DDT> : : =
                DDT
                | DTT<identifikace souboru>

<příkaz DUMP> : : = DUMP<identifikace souboru>

<příkaz ED> : : = ED<identifikace souboru>

<příkaz LOAD> : : =
                LOAD<identifikace souboru>
                | LOAD<jméno souboru>

<příkaz MOVCPM> : : =
                MOVCPM
                | MOVCPM<číslo>
                | MOVCPM *
                | MOVCPM<číslo> *
                | MOVCPM * *

<příkaz PIP> : : =
                PIP
                | PIP<výstup PIP> = <vstup PIP>

<příkaz STAT> : : =
                STAT
                | STAT VAL:
                | STAT<identifikace diskové jednotky>
                | STAT<identifikace skupiny souborů> <arg>
                | STAT<identifikace souboru> <atribut>
                | STAT<identifikace diskové jednotky> = R/O
```

| STAT DEV:
| STAT<seznam přiřazení>
| STAT<identifikace diskové jednotky>DSK:
| STAT USR:

<příkaz SUBMIT>: : =
SUBMIT<jméno procedury><parametry>

<příkaz SYSGEN> : : = SYSGEN

<jiný příkaz> : : = <jméno souboru><parametry>

<parametry> : : = <prázdný řetěz>
| <identifikace souboru><parametry>
| <řetěz><parametry>

<jméno procedury> : : = <jméno souboru>

<příkazová procedura> : : = <posloupnost příkazů>
| <příkaz XSUB><posloupnost příkazů>

<posloupnost příkazů> : : = <prázdný řádek>
| <příkaz><posloupnost příkazů>

<příkaz XSUB> : : = XSUB<konec>

<prázdný řádek> : : = <prázdný řetěz><konec>

<parametry ASM> : : =
<vstup ASM><výstup ASM><protokol ASM>

<vstup ASM> : : = <označení diskové jednotky>

<výstup ASM> : : = <označení diskové jednotky> | Z

<protokol ASM> : : = <označení diskové jednotky> | X | Z

<seznam vstupů PIP> : : =
<vstup PIP>
| <vstup PIP>, <seznam vstupů PIP>

<vstup PIP> : : =
<identifikace skupiny souborů><parametry PIP>
| <identifikace znakového zařízení><parametry PIP>
| INP: | EOF: | NUL:

<výstup PIP> : : =
<identifikace skupiny souborů> <parametry PIP>
| <identifikace znakového zařízení> <parametry PIP>
| OUT: | PRN:

<parametry PIP> : : = <prázdný řetěz>
| [<řetěz>]

<arg> : : = <prázdný řetěz> | $

<atribut> : : = $R/O | $R/W | $SYS | $DIR

<seznam přířazení> : : = <přiřazení>
| <přiřazení>, <seznam přiřazení>

<přiřazení> : : = <logické zařízení> = <fyzické zařízení>

<počet> : : = <číslo>
<identifikace souboru> : : =
<jednotka> <jméno souboru> . <typ souboru>

<jednotka> : : = <identifikace diskové jednotky>

<jméno souboru> : : = <řetěz 8>

<typ souboru> : : = <řetěz 3>

<identifikace skupiny souborů> : : =
<nejednoznačné jméno> . <nejednoznačný typ>

<nejednoznačné jméno> : : = <nejednoznačný řetěz 8>

<nejednoznačný typ> : : = <nejednoznačný řetěz 3>

<identifikace diskové jednotky> : : = <označení diskové jednotky> :
| <prázdný řetěz>
<označení diskové jednotky> : : =
A | B | C | D | E | F | G | H | I | J | K | L | M | N
| O | P
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p

<identifikace znakového zařízení> : : = <označení zařízení> :

<označení zařízení> : : = <označení logického zařízení>
| <označení fyzického zařízení>

<logické zařízení> : : = <označení logického zařízení> :

<fyzické zařízení> : : = <označení fyzického zařízení> :

<označení logického zařízení> : : = CON | RDR | PUN | LST

<označení fyzického zařízení> : : = TTY | CRT | BAT | UC1
| PTR | UR1 | UR2 | PTP | UP1 | UP2 | LPT | UL1

<číslo uživatele> : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11
| 12 | 13 | 14 | 15

<prázdný řetěz> : : =

<konec> : : = CR

<číslo> : : = <číslice>
| <číslice> <číslo>

<řetěz N> : : = <prázdný řetěz>
| <obyčejný znak> <řetěz N-1>

<nejednoznačný řetěz N> : : =
<prázdný řetěz>
| *
| <nejednoznačný znak> <nejednoznačný řetěz N-1>

<nejednoznačný znak>: : = <obyčejný znak> | ?

<řetěz> : : = <prázdný řetěz>
| <znak> <řetěz>

<znak> : : = <obyčejný znak> | <speciální znak>

<číslice> : : = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<obyčejný znak> : : = a | b | c | d | e | f | g | h | i | j | k | l | m
| n | o | p | r | s | t | u | v | w | x | y | z | A | B | C
| D | E | F | G | H | I | J | K | L | M | N | O | P
| R | S | T | U | V | W | X | Y | Z
| <číslice>
| + | - | ! | @ | # | $ | % | ^ | & | (|) | _ | ~
| ' | | | { | } | / | " | '

<speciální znak> : : = < | > | . | , | ; | : | = | ? | * | [|]

4.1.6 Speciální klávesy

Během činnosti programu pod operačním systémem CP/M má uživatel možnost použít ještě *speciálních kláves*. Nutnou podmínkou pro akceptování stisku speciální klávesy je samozřejmě komunikace se systémovou konzolí. Program, který komunikuje se systémovou konzolí je možno přerušit, je možno ovládat opis jeho výstupu na logickém zařízení pro tisk a příp. zadat konec vstupu pomocí následujících speciálních kláves.

Klávesa	Funkce
^C	Přerušení činnosti programu. Pokud program akceptuje vstup ze systémové konzole a ^C bylo stisknuto jako první znak v řádku, provede se teplý start systému, tj. znovu se zavedou moduly CCP a BDOS a řízení se předá procesoru příkazů. Pokud byla klávesa ^C stisknuta ne jako první znak na řádku, teplý start se neprovede, je zobrazeno ^C a předáno běžícímu programu.
^Z (EM)	Konec souboru (řetězu) při vstupu z klávesnice (např. ED, PIP).
^P	Řízení kontrolního opisu (hardcopy) – veškerý další výstup na systémovou konzoli (CON:) je současně kopírován též na logické zařízení pro tisk (LST:) až do dalšího stisku ^P nebo studeného startu systému.
^S	Řízení výstupu na systémovou konzoli. Při stisku ^S se pozastaví výstup na obrazovku až do následujícího stisku libovolné klávesy (klávesy ^Q u verze 3.0 – lze doporučit jako dobrou konvenci, která se používá i u jiných systémů).

4.1.7 Přehled příkazů

Příkazy pro procesor příkazů CCP systémů CP/M dělíme na zabudované a transientní. Zabudované příkazy provádí přímo procesor

příkazů, kód transientních příkazů musí být uložen na disku v souborech typu COM. Jsou to základní služební programy koncipované tak, aby mohly fungovat na libovolné instalaci systému CP/M (a jsou dodávány spolu se systémem).

Příkazy zabudované do CCP (rezidentní příkazy)

DIR [d:][afn]	zobrazení adresáře souborů
ERA afn	rušení souboru (souborů)
REN ufn1 = ufn2	přejmenování souboru
SAVE n ufn	úschova obsahu paměti na disk
TYPE ufn	znakové zobrazení obsahu souboru
d:	nastavení aktuální diskové jednotky
USER n	nastavení aktuálního uživatele

Standardní transientní příkazy

STAT	zobrazení a modifikace stavu systému,
PIP	univerzální kopírovací program,
SUBMIT	zpracování dávek příkazů,
XSUB	přesměrování vstupu pro dávku,
ED	textový editor,
DUMP	hexadecimální zobrazení obsahu souboru,
ASM	asembler pro procesor 8080,
LOAD	konverze výstupu ASM (HEX –> COM),
DDT	ladicí prostředek pro procesor 8080,
SYSGEN	záznam obrazu CP/M na disk,
MOVCPM	rekonfigurace CP/M.

Mimo tyto standardní příkazy obsahuje většina instalací CP/M celou řadu dalších programů, potřebných při práci se systémem, ale obvykle závislých na konkrétní instalaci; uveďme namátkou:

FORMAT	formátování disku,
TRANSFER	kopírování souborů přes paměť,
CONFI	programování stykových obvodů.

Pro systém CP/M existují též překladače většiny programovacích jazyků, např. Macro 80, Basic, PL/M, Pascal, ADA, Forth, Algol-60, Cobol, PL/1, C, Fortran, Lisp a další. K nim samozřejmě patří celá řada podpůrných programů (speciální editory, ladicí prostředky atd.). Velmi známý je rovněž textový systém WordStar, vybudovaný nad CP/M, a další systémy pro zpracování textu - Magic Wand, Microsoft Edit, Electric Pencil či SCOPE. Pod CP/M pracují i databázové systémy, např. dBASE-II, SELECTOR-IV, Pearl či HDMS. Pro systém CP/M byl vybudován i první tabulkový kalkulátor (spreadsheet) VisiCalc.

4.2. Příkazy zabudované v procesoru příkazů CCP

4.2.1 DIR - Výpis adresáře souborů

DIR [d:] [afn]

Příkaz DIR (DIRectory) zobrazí na systémové konzoli (logickém zařízení CON:) seznam jmen souborů, které se nacházejí na médiu v aktuální či uvedené (d:) diskové jednotce, přísluší aktuálnímu uživateli a patří do skupiny označené parametrem afn. Není-li parametr afn uveden, uvažují se všechny soubory aktuálního uživatele (totéž jako * . *). Do zobrazeného seznamu se nezahrnují jména souborů, které mají nastaven atribut SYS (viz popis služebního programu STAT). Tečka v identifikaci souboru se nezobrazuje. Pokud je adresář jednotky prázdný, nebo obsahuje pouze soubory s atributem SYS, není vypsáno nic.

Příklady použití:

DIR	Zobrazí jména a typ všech souborů aktuálního uživatele na médiu v aktuální diskové jednotce (ekvivalentní DIR * . *).
DIR d:	Totéž, ale pro určenou diskovou jednotku.
DIR filename.typ	Adresář bude obsahovat pouze jeden soubor. Pokud neexistuje, ohlásí se zpráva NO FILE.

DIR *.typ	Zobrazí adresář všech souborů uvedeného typu na aktuální diskové jednotce.
DIR filename.*	Zobrazí adresář všech typů uvedeného souboru na aktuální diskové jednotce.
DIR A*.*	Zobrazí adresář všech souborů, jejichž jméno začíná písmenem A na aktuální diskové jednotce.
DIR A???????.???	Totéž
DIR A??.*	Totéž, ale pouze soubory, jejichž jméno není delší než 3 znaky.

4.2.2 ERA – Zrušení souboru (skupiny souborů)

ERA [d:] afn

Příkaz ERA (ERAse) zruší udanou skupinu souborů (afn) aktuálního uživatele na médiu v aktuální či udané (d:) diskové jednotce. (Zapíše kombinaci 0E5H do první slabiky položek adresáře, které přísluší udaným souborům – existuje proto možnost restaurovat zrušené soubory bezprostředně po jejich zrušení).

Příklady použití:

ERA filename.typ	Zruší soubor filename.typ v adresáři aktuálního uživatele na aktuální diskové jednotce (ale nikoli na jiných diskových jednotkách).
ERA *.*	Zruší všechny soubory aktuálního uživatele na aktuální diskové jednotce; v tomto případě je zrušení souborů (souboru) provedeno až po vyžádaném potvrzení zprávou ALL (Y/N)?
ERA filename.*	Zruší všechny typy uvedeného souboru v adresáři aktuálního uživatele na aktuální diskové jednotce.
ERA d:*.typ	Zruší všechny soubory uvedeného typu v adresáři aktuálního uživatele na specifikované (d:) diskové jednotce.

Poznámka: ve víceuživatelské variantě MP/M existuje navíc příkaz ERAQ (ERAse with Query), který vyžaduje potvrzení před zrušením každého souboru.

4.2.3 REN – Přejmenování souboru

REN ufn1 = ufn2

Příkaz REN (RENname) přejmenuje soubor ufn2 na ufn1. Pokud soubor ufn1 již existuje, ohlásí chybu FILE EXIST a změnu neprovede. Přejmenování neznamená kopírování souboru, a proto je lze uskutečnit pouze v rámci jedné diskové jednotky. Označení diskové jednotky může být uvedeno v prvém (ufn1), ve druhém (ufn2) či v obou paramterech, platí pro oba případy. Je-li uvedeno v obou parametrech, musí být stejné. V parametrech nelze použít znaky " ? " a " * ", tzn. že není možno najednou přejmenovat celou skupinu souborů.

Příklady použití:

REN newname.ntp=oldname.otp	Přejmenuje soubor oldname.otp na newname.ntp v rámci aktuální diskové jednotky v oblasti aktuálního uživatele.
REN B:X.Y = X.BAK	Přejmenuje soubor X.BAK na jednotce B: v oblasti aktuálního uživatele.
REN X.Y = B:X.BAK	Totéž.
REN B:X.Y = B:X.BAK	Totéž.

Poznámka: v příkazu REN je dodržena standardní konvence systému CP/M, která se týká používání rovnítka a kterou lze vyjádřit zápisem

VÝSTUP = VSTUP (příp. VSTUPY).

Je zde patrna snaha připodobnit tvar zápisu parametrů tvaru přiřazovacího příkazu vyšších programovacích jazyků.

4.2.4 SAVE – Uložení obsahu paměti do souboru

SAVE n ufn

Příkaz SAVE vytvoří v adresáři aktuálního uživatele soubor ufn obsahující kopii n (dekadicky) stránek (bloků délky 256 slabik) souvisle od počátku TPA (obvykle od adresy 100H). Program, uložený tímto způsobem do souboru typu COM, lze pak opět zavést do paměti a spustit pomocí procesoru příkazů.

Příklad použití:

SAVE 10 B:PROG.COM Uloží kopii 10 stránek (tj. obsah operační paměti v rozsahu od adresy 100H do adresy 0AFFH) do souboru PROG.COM v oblasti aktuálního uživatele na diskové jednotce B (pozor: 0AH = 10 dekadicky).

Poznámka: Pokud soubor uvedený v příkazu SAVE již existoval, je přepsán novým obsahem. Lze proto doporučit použití DIR před příkazem SAVE. Příkaz SAVE se vymyká z úrovně ostatních zabudovaných příkazů a bývá používán v kombinaci s ladicími prostředky (např. DDT) pro modifikaci programu na úrovni strojového kódu či asembleru.

4.2.5 TYPE – Znakové zobrazení obsahu souboru

TYPE ufn

Příkaz TYPE zobrazí na obrazovce obsah souboru ufn. Předpokládá se, že soubor obsahuje text, tj. obsah souboru ufn se interpretuje jako posloupnost znaků v kódu ASCII a beze změn se kopíruje na systémovou konzoli (pro prohlížení netextových souborů viz DUMP). Interpretovány jsou pouze znaky CR a LF jako přechod na nový řádek a tabulátory jsou nahrazeny příslušným počtem mezer.

Poznámka: Zobrazení lze pozastavit stiskem ^S a opět spustit ^Q (u verze 2.x a nižších libovolným znakem, i např. ^S). Výstup je možno

přerušit stiskem libovolného jiného znaku (např. ^C). Výstup je ukončen při nalezení znaku ^Z (kód 1AH) ve vypisovaném souboru nebo jeho vyčerpáním (pokud je délka souboru celistvým násobkem délky sektoru). Toho je možno využít k zabránění výpisu celého souboru, uvedeme-li znak ^Z před chráněnou částí textu. Např.:

```
(C) XXX 1985 ^Z
chráněný text.
^Z
```

Příklad použití:

```
TYPE HELP.TXT
```

Poznámka: Příkaz TYPE používáme obvykle k výpisu souborů typu TXT, DOC, HEX, BAK, PRN, SRC, ASM atd.

4.2.6 Nastavení aktuální diskové jednotky

```
X:
```

Uvedená jednotka X se stane aktuální. Systém zaznamená aktuální jednotku do komunikační zóny (SPA) a využívá ji jako implicitní v těch případech, kdy není v příkazech explicitně uvedena jednotka jiná. Aktuální disková jednotka je zobrazena ve výzvě systému. Systém dovoluje spolupráci až se šestnácti diskovými jednotkami, povolená označení X diskové jednotky jsou tedy A až P (u verzí nižších než 2.x jsou povoleny pouze čtyři diskové jednotky, označené A až D).

Příklad použití:

```
A>B          (změníme aktuální jednotku)
B>TYPE X.Y   (ve výzvě je zobrazena nová jednotka)
.........    (obsah souboru B: X.Y)
B> .
```

Poznámka: nastavení akutální jednotky platí až do další změny či studeného startu systému. Aktuální disková jednotka funguje jako implicitní,

ale moduly CCP ą BDOS jsou vždy zaváděny ze systémové jednotky A. Při tepelném startu (ukončení programu) proto systém vždy čte jak ze systémové jednotky (moduly CCP a BDOS), tak z aktuální jednotky (adresář). Aktuální disková jednotka se proto stává současně aktivní jednotkou, neboť adresář založeného média byl přenesen do operační paměti a je zde dostupný pro další přístup k médiu přes tabulky systému.

4.2.7 USER – Nastavení aktuálního uživatele

USER n

Příkaz USER nastaví aktuální uživatele určeného parametrem n (dekadicky). Slouží zejména pro sdílení pevných disků více uživateli. Povolená označení uživatelů jsou 0 až 15. Po zavedení systému (studeném startu) je nastaven jako aktuální uživatel 0. Každý soubor na diskovém médiu má v adresáři zaznamenáno označení uživatele. Diskové médium je tím virtuálně rozděleno na jednotlivé uživatelské oblasti a každý uživatel má běžně přístup pouze do své oblasti (kterou si ale může nastavit libovolně). Informace o aktuálním uživateli a použitých oblastech na disku je možno získat pomocí služebního programu STAT.

Příklad použití:

```
56K CP/M VERSION 2.2 (studený start systému – nastaven uživatel 0)
A>DIR                (zobrazíme adresář uživatele 0)
A: PROGR COM: PROGR SRC
A>USER 2             (nastavíme nového uživatele)
A>DIR                (výpis adresáře uživatele 2)
A: ALFA TXT : BETA BAK : GAMA  COM
A>ERA *.*            (zrušíme všechny soubory v oblasti uživatele 2)
ALL (Y/N)? Y
A>DIR
NO FILE
A>USER 0             (vrátíme se k uživateli 0)
A>DIR                (jeho oblast zůstala nedotčena)
A:PROGR COM : PROGR SRC
```

4.2.8 Spuštění programu

[jednotka:] jméno [_ parametry]

Procesor příkazů CCP hledá na aktuálním či udaném disku soubor jméno COM, který zavede do paměti (od adresy 100H). Případně parametry procesor příkazů CCP předzpracuje a zapíše do komunikační zóny (SPA) a poté předá řízení zavedenému programu (na adresu 100H).

Poznámky: Případné parametry pro program musí být odděleny mezerou. Procesor příkazů CCP předává řízení programu instrukcí CALL 100H – toho lze využít v případech, kdy spuštěný program nepřepisuje procesor příkazů CCP (a samozřejmě ani jinou část systému) k návratu do procesoru příkazů CCP instrukcí RET, namísto standardního návratu instrukcí JMP 0.

4.3 Služební programy systému CP/M

Firma Digital Research dodává v distribuční verzi systému CP/M následující sadu služebních programů, které se též nazývají transientní příkazy, neboť jejich prováděcí kód musí být uložen běžným způsobem v samostatných souborech na diskovém médiu.

STAT	zobrazení a modifikace stavu systému
PIP	univerzální kopírovací program
ED	textový editor
SUBMIT	řízení zpracování dávek příkazů
XSUB	
DUMP	hexadecimální výpis obsahu souboru
ASM	asembler pro 8080
LOAD	konverze HEX —> COM
DDT	ladicí prostředek pro 8080
SYSGEN	záznam CP/M na disk
MOVCPM	rekonfigurace CP/M

4.3.1 STAT – Zobrazení a modifikace stavu systému

STAT [parametry]

Program STAT (STATus) slouží pro zobrazení a modifikace základních stavových údajů systému CP/M. Pomocí programu STAT lze získat údaje o obsazení diskových médií, rozsahu souborů, charakteristik diskových jednotek a aktuálním uživateli; dále pro zobrazení či modifikaci statutu diskových jednotek a souborů. Program STAT rovněž umožňuje zobrazení aktuálního přiřazení logických a fyzických zařízení a jeho modifikaci. Přehled všech možností, které poskytuje program STAT, získáme speciálním tvarem příkazu.

Zobrazení repertoáru možností STAT

Tvar příkazu:	STAT VAL:
Temp R/O Disk :	d: = R/O
Set Indicator :	d: filename.typ $R/O $R/W $SYS $DIR
Disk Status :	DSK: d: DSK:
User Status :	USR:
Iobyte Assign :	

CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:

Použití STAT pro získání informací o souborech

STAT Zobrazí statut a rozsah volného prostoru pro všechny aktivní diskové jednotky ve tvaru:

d: R/W, SPACE : nnnk nebo
d: R/O, SPACE : nnnk,

kde R/W (Read and Write), R/O (Read Only) je statut jednotky a nnn je volný prostor na médiu v KB.

STAT d: Zobrazí volný prostor na disku d: ve tvaru:

BYTES REMAINING ON d: nnn k

STAT afn Zobrazí statistické údaje o souboru (nebo souborech, pokud není jméno jednoznačné) ve tvaru:

Recs Bytes Ext Acc
rrrr nnnnK ee aaa d: filename.typ (pro každý soubor),

kde rrrr je počet sektorů (128 slabik), které soubor zabírá na disku,
nnnn je velikost souboru v KB (1024 slabik)
ee je počet rozšíření souboru (16 KB)
aaa jsou přístupová práva (R/W, R/O, SYS – viz dále)
d: filename.typ je jméno souboru (pokud se jedná o soubor s atributem SYS, je uvedeno v závorkách)

Na závěr program STAT vypíše informaci o volném místě na disku:

BYTES REMAINING ON d: nnnn k .

Uvedeme-li za identifikací souboru (skupiny) $S, bude navíc zobrazen počet virtuálních sektorů alokovaných pro soubor. U sekvenčních souborů je počet alokovaných a použitých sektorů stejný, ale u souborů s přímým přístupem to tak být nemusí. Tvar:

Size Recs Bytes Ex Acc
ssss rrrr nnnnK ee aaa d: filename.typ . . . ,

kde ssss je počet virtuálních sektorů alokovaných pro soubor.

Použití STAT pro ovládání statutu diskových jednotek a atributů souborů

Struktura uložení informace na disku v rámci CP/M dovoluje (pro verze 2.x a vyšší) nastavit u každého souboru několik příznaků, které mohou nést různou informaci o souborech. Dva z těchto příznaků jsou využity pro reprezentaci atributů souborů:
– příznak, že soubor může být pouze čten nebo jej lze i přepsat – přístupová práva R/O (Read/Only), R/W (Read/Write)

– příznak, že soubor je běžný (DIR) nebo systémový (SYS), tj. nelze jej kopírovat a je vypuštěn z výpisu adresáře. Jednotlivé atributy souboru lze nastavit pomocí příkazů: ($ slouží jako omezovač jména souboru)

STAT ufn $R/O	Soubor ufn je možno pouze číst,
STAT ufn $R/W	Soubor ufn je možno přepisovat i rušit,
STAT ufn $SYS	Soubor není možno kopírovat a není zobrazen ve výpisu adresáře,
STAT ufn $DIR	Soubor je zobrazen ve výpisu adresáře a je možno jej kopírovat.

Výše uvedené příznaky jsou zaznamenány v adresáři na disku a platí proto trvale – mimoto lze pracovně zakázat zápis na celou diskovou jednotku příkazem:

STAT d: = R/O.

Toto nastavení se nezaznamenává na disk (k tomu slouží mechanická indikace na médiu) a při teplém či studeném startu systému jsou všechny aktivní diskové jednotky nastaveny jako R/W.

Poznámka: Při výměně média systém automaticky označí disk atributem R/O, aby nedošlo k nesprávnému přepsání informace a při pokusu o zápis je hlášeno

BDOS ERROR ON d: READ ONLY,

pokud se pokoušíme zapisovat nebo zrušit soubor s atributem R/O nebo soubor na disku se statutem R/O. Pomocí speciálních klíčů lze programem PIP přepisovat i soubory s atributem R/O a kopírovat i soubory atributem SYS (viz popis PIP).

Použití STAT pro ovládání přiřazení fyzických zařízení logickým

Aktuální přiřazení fyzických zařízení logickým je možné zobrazit příkazem:

STAT DEV:

Zobrazení může např. být:

CON: = CRT:
RDR: = PTR:
PUN: = PTP:
LST: = LPT:

Danému logickému zařízení můžeme přiřadit jen určitá fyzická zařízení. Která to mohou být, zjistíme příkazem STAT VAL:, který zobrazí nabídku možností:

Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:

Aktuální přiřazení logických a fyzických zařízení můžeme změnit (nastavit) příkazem:

STAT log: = fyz: nebo
STAT lo1: = fy1:, lo2: = fy2:, ... ,

samozřejmě při respektování správných možností (viz STAT VAL: např. logickému zařízení LST: můžeme přiřadit pouze fyzická zařízení TTY: nebo CRT: nebo LPT: nebo UL1:).

Poznámka: Aktuální přiřazení logických a fyzických zařízení je uloženo v tzv. IOBYTE (pouze verze 2.x a nižší!), jehož stav můžeme číst a měnit pomocí funkcí BDOS (viz popis BDOS). Rovněž spolupráce s logickými zařízeními je možná pomocí funkcí BDOS, který je převádí na volání funkcí BIOS. Při instalaci systému CP/M je definováno, jaká zařízení a jakým způsobem BIOS ovládá (viz popis BIOS).

Použití STAT pro zobrazení charakteristik disku

STAT DSK: nebo STAT d:DSK:

Program STAT zobrazí informace o disku např. v následujícím tvaru:

```
    B : Drive Characteristics
65536 : 128 Byte Record Capacity
 8192 : Kilobyte Drive Capacity
  128 : 32 Byte Directory Entries
    0 : Checked Directory Entries
 1024 : Records/Extent
  128 : Records/Block
   58 : Sectors/Track
    2 : Reserved Tracks
```

Příklad ukazuje charakteristiky pevného disku v jednotce B, který má kapacitu 65536 sektorů délky 128 slabik, tj. o celkové kapacitě 8 MB (8192 KB). Disk může obsahovat až 128 souborů, neboť adresář média má 128 položek. Z počtu hlídaných položek v adresáři (zde 0) je poznat, že se jedná o pevný disk. Obvykle je počet hlídaných položek roven počtu položek adresáře, neboť tímto způsobem systém CP/M hlídá případnou výměnu média (před zápisem na disk je porovnán adresář na disku s adresářem v paměti). U pevného disku je hlídání výměny média zbytečné.

Stopy disku jsou rozděleny do 58 sektorů. Alokační blok obsahuje 128 sektorů, tj. jednou položkou adresáře můžeme adresovat 1024 sektorů (extent). Počet rezervovaných stop udává počátek adresáře na disku (lze využít pro simulaci více virtuálních disků na jednom médiu).

Použití STAT pro zobrazení aktuálního uživatele:

```
STAT USR:
```

STAT zobrazí informace o aktuálním uživateli a o tom, kteří uživatelé používají aktuální disk, např. ve tvaru:

```
Active User  : 0
Active Files : 0 1 3,
```

což znamená, že aktuální uživatel má číslo 0 a na aktuálním disku mají též soubory uživatelé 1 a 3.

Přehled funkcí programu STAT:

STAT	Zobrazí statut aktivních diskových jednotek a volný prostor.
STAT d:	Totéž, ale pro označenou jednotku.
STAT ufn	Zobrazí informace o souboru ufn.
STAT ufn $S	Totéž, navíc počet alokovaných sektorů.
STAT afn	Totéž, ale pro skupinu souborů afn.
STAT VAL:	Zobrazí přehled možných parametrů STAT pro nastavení statutu a přiřazení logických a fyzických zařízení.
STAT d: =R/O	Nastaví statut diskové jednotky R/O – povoleno pouze čtení.
STAT ufn $R/O	Nastaví statut (přístupová práva) souboru R/O – povoleno pouze čtení.
STAT ufn $R/W	Totéž, ale R/W – povoleno čtení i zápis.
STAT ufn $SYS	Totéž, ale SYS – systémový soubor (není zobrazován DIR).
STAT ufn $DIR	Totéž, ale DIR – běžný soubor (je zobrazován příkazem DIR).
STAT DEV:	Zobrazí aktuální přiřazení logických a fyzických zařízení.
STAT log: = fyz:	Přiřadí fyzické zařízení fyz: logickému zařízení log:.
STAT DSK:	Zobrazí charakteristiky disku.
STAT USR:	Zobrazí aktuálního uživatele a všechny uživatele disku.

4.3.2 SUBMIT, XSUB – Zpracování příkazových procedur

SUBMIT ufn [_ parametr] *

Program SUBMIT vytváří na základě příkazové procedury uložené v souboru ufn dávku příkazů uloženou v souboru $$$.SUB na aktuální diskové jednotce. Soubor ufn musí být povinně typu SUB a přípona .SUB se v příkazu SUBMIT neuvádí.

Dávka může obsahovat formální parametry označené $1, $2, ... , $9. V okamžiku zahájení zpracování příkazu SUBMIT jsou za tyto formální parametry dosazeny skutečné parametry uvedené v příkazu SUBMIT za jménem příkazové procedury. Náhrada parametrů se provádí textově

v odpovídajícím pořadí (první parametr za $1 atd.). Skutečné parametry jsou v příkazu SUBMIT odděleny mezerou. Nesouhlas počtu formálních a skutečných parametrů způsobí ukončení provádění příkazu SUBMIT a je indikováno chybovou zprávou.

Příkaz SUBMIT vytváří na aktuální diskové jednotce soubor označený $$$.SUB, obsahující příkazy ze souboru ufn.SUB po substituci odpovídajících parametrů. Tvar souboru $$$.SUB je ovlivněn jednoduchým zpracováním. Řádky souboru ufn.SUB jsou uloženy v obráceném pořadí, aby po zpracování příkazu bylo možno jednoduše řádek vypustit pouhým zkrácením souboru $$$.SUB. Jednoduché zkrácení je umožněno tím, že každý řádek je v souboru $$$.SUB uložen v jedné logické větě (délky 128 slabik). Zkrácení se pak provede oddělením poslední věty. Délka dávky příkazů $$$.SUB je omezena na maximálně 128 příkazů.

Po ukončení zpracování příkazu SUBMIT je řízení předáno systému, který aktivuje procesor příkazů. Pokud je aktuální jednotka současně systémovou jednotkou, čte procesor příkazů CCP příkazy nejprve z takto vytvořeného souboru A:$$$.SUB. Po vyčerpání obsahu souboru A:$$$.SUB (tzn. po provedení všech příkazů nebo nalezení prázdného řádku) je soubor procesorem příkazů zrušen. Pokud CCP zjistí chybu v některém příkazu dávky, je zpracování dávky ukončeno. Uživatel může zpracování dávky přerušit stiskem klávesy RUBOUT, které je akceptováno v okamžiku přechodu na další příkaz dávky. Zpracování dávky není přerušeno chybou indikovanou v programu (např. chybou při překladu). Pokud to charakter zpracování vyžaduje, je nutné, aby příslušný program sám zrušil soubor A:$$$.SUB.

V kombinaci s programem XSUB.COM lze mezi systémové příkazy v dávce vkládat i vstupní údaje pro programy (např. příkazy pro DDT atd.). V tom případě musí být jako první příkaz dávky uveden příkaz XSUB, který modifikuje systémovou funkci pro čtení řádku ze systémové konzole (funkce 10 modulu BDOS) a přesměruje ji na soubor A:$$$.SUB. Program XSUB je v činnosti během zpracování dávky a jeho přítomnost je indikována zprávou (XSUB ACTIVE). Program XSUB nelze použít pro přesměrování vstupu programů, které pracují s klávesnicí jiným způsobem.

Poznámky: Dávka nesmí obsahovat prázdný řádek, neboť v tom případě je zpracování dávky bez provádění ukončeno. Souvisí to se způsobem uložení příkazů v souboru $$$.SUB, kde jsou příkazy uloženy v obráceném

pořadí a prázdný řádek procesor příkazů CCP využívá jako příznak konce dávky, tj. příznak pro zrušení souboru A:$$$.SUB.

Chceme-li použít znak "$" v příkazech dávky, musíme jej zdvojit (např. X.$$$ zapíšeme jako X.$$$$$$).

Chceme-li v dávce použít řídicí znaky (^X), zapíšeme je jako dvojici znak "^" a znak "X" .

Dávka může obsahovat další příkaz SUBMIT – řetězení dávek (viz příklad), který však ukončí zpracování aktuální dávky (přepíše soubor A:$$$.SUB).

Příklady použití SUBMIT:

– Použití SUBMIT pro zjednodušené zadávání komplikovaných a často používaných příkazů:

Tvar dávky (obsah souboru COMPOSE.SUB):

```
PIP TEXT.TXT=PART1.TXT,PART2.TXT,PART3.TXT
```

Tvar příkazu: SUBMIT COMPOSE

– Použití SUBMIT při opakované činnosti:

Tvar dávky (obsah souboru JOB.SUB):

```
ED $1.ASM
ASM $1
DIR $1.*
ERA $1.BAK
PIP CON: = $1.PRN
ERA $1.PRN
LOAD $1
DDT $1.COM
SUBMIT JOB$1
```

Tvar příkazu: SUBMIT JOB SOURCE

Provádí v cyklu opravu, překlad, výpis, ladění programu SOURCE. Cyklus ukončíme stiskem klávesy RUBOUT.

– Použití SUBMIT pro přípravu rutinních činností:

Tvar dávky (obsah souboru COPYSYS.SUB).

```
XSUB
PIP
$2:=$1:STAT.COM[R]
$2:=$1:PIP.COM[R]
$2:=$1:SUBMIT.COM[R]
$2:=$1:XSUB.COM[R]
$2:=$1:MESSAGE.TXT[R]
CON:=$1:MESSAGE.TXT[R]
```

Tvar příkazu: SUBMIT COPYSYS A B

Zkopíruje uvedené soubory z A: na B: a na konzoli zobrazí obsah souboru MESSAGE.TXT. V tomto souboru může být uložena výzva uživateli pro ukončení dávky COPYSYS stiskem klávesy CR, neboť odpovídající prázdný řádek nelze do dávky vložit (viz poznámky).

– Vytvoření disku s automatickým odstartováním programu např. po výpadku napájení:

Vytvoříme např. na diskové jednotce B: soubor B:$$$.SUB, obsahující jediný příkaz TELEX. Aktuální diskovou jednotkou je A:. Po vložení takto připraveného disku do jednotky A: a zavedení systému, procesor příkazů CCP automaticky spustí program TELEX.COM, který zaznamenává přicházející telexy na disk. Po případném výpadku napájení je znovu automaticky spuštěn program TELEX, příp. bez nutnosti přítomnosti operátora. Pokud ale předá program TELEX řízení systému, procesor příkazů CCP zruší soubor A:$$$.SUB (protože již byl proveden) a automatické spuštění je zničeno.

4.3.3 PIP – Univerzální kopírovací program

```
PIP [ _ kam=odkud [,odkud]*]
```

Program PIP (Peripheral Interchange Program) slouží pro přenos souborů mezi periferními jednotkami. Parametry udávají odkud a kam se má soubor (soubory) přenášet. Pokud v příkazu nejsou uvedeny

parametry, jsou vyžadovány na uživateli výzvou "*", po níž uživatel zadává paramtery ve tvaru:

kam=odkud[,odkud]*.

V tomto případě je možno použít PIP vícekrát bez nutnosti opětovného zavádění do paměti. Odpovíme-li na výzvu "*" stiskem klávesy CR (RETURN, atd – klávesy s kódem 13), vrátí program PIP řízení systému. Povolená označení kam a odkud jsou popsána dále. Uvedeme-li více označení odkud, je výstup spojením všech vstupů. Kopírování lze dále řídit pomocí tzv. *přepínačů* , jejichž přehled je uveden dále.

Poznámka: Označení kam a odkud se musí lišit. V případě, kdy kam označuje již existující soubor, je tento přepsán bez zprávy uživateli.

Povolená označení výstupu pro program PIP

Jako výstup pro kopírování můžeme označit libovolný soubor, logické nebo fyzické zařízení, které je schopno soubor přijmout. Povolená *označení výstupu* (parametr kam) jsou:

Jednotka	Disková jednotka
ufn	Jméno souboru
afn	Znaky "?" a "*" jsou nahrazeny odpovídajícími znaky či položkami v označení vstupu
CON:	Systémová konzole
CRT:	Rychlá zobrazovací jednotka
UC1:	Uživatelsky definovaná konzole
LST:	Logické zařízení pro tisk
LPT:	Fyzické zařízení tiskárna
UL1:	Uživatelsky definovaná tiskárna
TTY:	Dálnopis
PRN:	Totéž jako LST:, ale expanduje tabelátory, čísluje řádky a automaticky stránkuje po 60 řádcích
PUN:	Logické zařízení znakového výstupu
PTP:	Děrovač pásky
UP1:	Uživatelsky definovaný děrovač 1
UP2:	Uživatelsky definovaný děrovač 2

OUT:	Speciální výstup, který musí být zaprogramován do PIP např. pomocí DDT (viz popis). Způsobí, že výstup každého znaku je interpretován jako volání podprogramu na adrese 106H, přenášený znak je uložen v registru C. Lze použít pro výstup na speciální zařízení (např. zapisovač).

Povolená označení vstupu pro program PIP

Jako vstup na kopírovací program PIP můžeme označit libovolný soubor, logické nebo fyzické zařízení, z nějž lze soubor číst. Navíc lze využít dvou pseudozařízení. Povolená *označení vstupu* (parametr odkud) jsou:

Jednotka	Disková jednotka
ufn	Jméno souboru
afn	Označení skupiny souborů (v tomto případě se kopírují všechny soubory patřící do skupiny jako samostatné soubory – nespojují se)
CON:	Systémová konzole
TTY:	Dálnopis
CRT:	Rychlá konzole
UC1:	Uživatelsky definovaná konzole
RDR:	Logické zařízení znakového vstupu
PTR:	Snímač děrné pásky
UR1:	Uživatelsky definovaný snímač 1
UR2:	Uživatelsky definovaný snímač 2
IPN	Speciální vstup, který musí být zaprogramován do PIP např. pomocí DDT (viz popis). Způsobí, že vstup znaků je interpretován jako volání podprogramu na adrese 103H, který vrátí přečtený znak na adrese 109H (s nulovým paritním bitem). Lze použít pro vstup ze speciálního zařízení (např. tableta).
NUL:	Pseudozařízení – představuje 40 prázdných znaků (blank). Lze použít při výstupu děrné pásky
EOF:	Pseudozařízení – představuje znak konce souboru (^Z).

Přepínače programu PIP

Kopírování programem PIP lze dále řídit pomocí přepínačů. *Přepínače* uvádíme za jménem souboru (skupiny souborů), k němuž se vztahují, uzavřené mezi hranaté závorky []. Jsou povoleny kombinace přepínačů – jednotlivé přepínače můžeme psát bezprostředně za sebou nebo oddělit jednou či více mezerami. Např. [NPE] znamená totéž co [N P E]. Možné přepínače uvádíme včetně hranatých závorek a v kulatých závorkách uvádíme mnemotechniku.

[B] (Block mode): Čte spojitě do paměti až po výskyt znaku "x-off" (CTRL–S). Poté zapíše obsah paměti na výstup a opakuje čtení atd. Má smysl pro zařízení nepracující ve start/stopním režimu (např. kazetový magnetofon). Při přeplnění paměti během čtení hlásí chybu.

[Dn] (Delete): Kopíruje pouze úvodních n znaků na řádku, zbytek řádku nepřenáší. Slouží pro tisk "širokých" souborů na "úzkém" zařízení.

[E] (Echo): Zobrazuje vystupující informaci též na konzoli (obrazovce). Lze použít pro kontrolu co se kopíruje.

[F] (Formfeeds removed): Vynechává ze vstupu všechny znaky FF (ASCII) tj. OCH.

[Gn] (Get from user n): Povoluje čtení souboru z oblasti uživatele n. Nelze použít pro zápis do oblasti jiného než akutálního uživatele.

[H] (Hex file): Při kopírování je prováděna kontrola, zda vstup obsahuje pouze platný formát HEX firmy Intel. Při nalezení jiného znaku je vyvolána interakce s operátorem. Slouží pro kontrolu přenosu souboru typu HEX (výstup ASM).

[I] (Ignore zeros): Při kopírování vynechá ze vstupu záznamy ":00" formátu HEX. Implicitně nastaví parametr H.

[L] (Lower case): Převádí velké znaky na malé.

[N] (Number lines): Čísluje řádky na výstupu s potlačením vedoucích nul v číslování. Číslování je prováděno s krokem 1 od 1. Za číslo řádky je vložena dvojtečka.

[N2] Totéž jako N, ale v číslování nejsou potlačeny vedoucí nuly a za číslo řádku je vložen tabelátor (09H CTRL–I).

[O] (Object file): Ignoruje CP/M označení konce souboru – znak ^Z. Slouží pro kopírování netextových souborů a ignorování znaků konce souboru při spojování souborů.

[P] (Page): Vkládá znak nová stránka (FF–ASCII, OCH) před první řádek a za každý 60. řádek. Slouží pro kopírování na tiskárnu.

[Pn] Totéž jako P, ale za každý n-tý řádek.

[Qstring^Z] (Quit): Zastaví kopírování před prvním výskytem řetězu string (který musí být v zápisu přepínače ukončen znakem ^Z). Chceme-li v řetězu použít malá písmena, je nutno obejít vstup přes CCP (který vždy konvertuje malá písmena na velká) a využít volání PIP bez parametrů. Po výzvě " * " lze zadávat i malá písmena (PIP nekonvertuje malá písmena na velká).

[R] (Read system file): Povolí kopírování souboru s příznakem SYS. Kopie bude mít rovněž příznak SYS.

[Sstring^Z] (Start): Obnoví kopírování (viz přepínač Q) na prvním výskytu řetězu string (včetně). Lze též použít pro pokračování v kopírování např. po výpadku tiskárny.

[Tn] (Tabulate): Expanduje tabelátory (09H, ^I) na každý n-tý sloupec.

[U] (Upper case): Převádí malé znaky na velké.

[V] (Verify): Ověření zapsané informace kontrolním přečtením.

[W] (Write-over): Povolí přepsání souboru chráněného proti zápisu (R/O). Není-li uveden, pak snaha přepsat chráněný soubor vyvolá dotaz na operátora:

DESTINATION FILE IS R/O, DELETE (Y/N)?

Pro odpovědi Y je soubor přepsán, jinak je kopírování přerušeno se zprávou:

NOT DELETED

Parametrem W tuto konverzaci vynecháme. Parametr W je určen pro výstup, ale uvádí se u posledního vstupního souboru!

[X] Vypne kontrolu na platné znaky kódu ASCII, která je jinak prováděna. Slouží pro kopírování souboru v jiném kódu.

[Z] (Zero parity bit): Nuluje ve všech znacích paritní bit. Použitelné např. pro konverzi dokumentů pořízených programem WORDSTAR, který využívá paritní bity pro doplňkové informace.

Poznámka: Příkaz PIP LST:=ufn[N T8 P60] je ekvivalentní příkazu PIP PRN:=ufn.

Příklady použití programu PIP

Uvádíme pouze parametry, které mohou být uvedeny po příkazu PIP nebo výzvě "*".

A:=B:filename.typ	Kopíruje soubor filename.typ z jednotky B: na jednotku A: (A:filename.typ=B: znamená totéž).
newname.*=oldname.typ	Kopíruje soubor oldname.typ do souboru newname.typ na aktuální diskové jednotce.
*.BAK=filename.typ	Kopíruje soubor filename.typ do souboru filename.BAK na aktuální diskové jednotce.
A:=B:*.*	Kopíruje všechny soubory se statutem DIR aktuálního uživatele z diskové jednotky B: na A:.
A:=B:*.*[V]	Totéž, ale s verifikací.
A:=B:*.*[R V]	Totéž, ale i soubory se statutem SYS.
A:=B:filename.*	Kopíruje všechny soubory (DIR) se jménem filename z diskové jednotky B: na A:.
A:=B:*.typ	Totéž, ale všechny soubory uvedeného typu.
filename.typ=CON	Kopíruje vstup z klávesnice do souboru filename.typ. Vstup je ukončen znakem ^Z.
PUN:=NUL:,filename.typ,NUL:	Vyděruje soubor s úvodní a zakončovací mezerou na pásce.
filename.typ=RDR:	Přečte soubor z děrné pásky.
LST:=filename.typ	Vytiskne soubor.
newname.typ=a1.typ,a2.typ,a3.typ	Vytvoří soubor newname.typ spojením textových souborů a1.typ, a2.typ a a3.typ
newname.typ=a1.typ[X],a2.typ[X]	Spojí netextové soubory.
CHAPTER3.TXT=VOLUME.TXT [SCHAPTER3^Z QCHAPTER^Z]	Zkopíruje text ze souboru VOLUME.TXT mezi řetězy CHAPTER3 (včetně) a CHAPTER4 (nekopíruje).

A:=A:*.*[G2]	Kopíruje všechny soubory z uživatelské oblasti 2 do uživatelské oblasti aktuálního uživatele.
A:=B:*.COM[W]	Zkopíruje soubory typu COM z B: na A:. Původní soubory bez ohledu na ochranu přepíše!

Poznámka: pro zkopírování vlastního programu PIP z uživatelské oblasti 0 do uživatelské oblasti 3 je nutno použít posloupnost:

A>USER 0	(uživatel 0)
A>DDT PIP.COM	(natáhne PIP do paměti)
(nutno přečíst velikost n – délku programu PIP)	
-G0	(ukončení DDT)
A>USER 3	(uživatel 3)
A>SAVE n PIP.COM	(uloží PIP na disk)

4.3.4 ED – Textový editor

ED_ufn

V každém uživatelském prostředí je obvykle zapotřebí program pro přípravu a opravy textových souborů. Schopnosti a způsob používání textového editoru jsou obvykle do značné míry limitovány schopnostmi systémové konzole. Vzhledem k rozmanitosti systémových konzolí na různých instalacích systému CP/M, dodává firma Digital Research textový editor ED, který umožňuje přípravu a úpravu textových souborů na libovolné instalaci systému CP/M. Na druhé straně je tím omezen uživatelský komfort editoru ED, který pracuje jako znakový editor.

Pokud editor ED najde soubor ufn, otevře jej pro čtení. Není-li nalezen, informuje zprávou NEW FILE, že bude vytvářen nový soubor. Dále otevře pracovní soubor d:filename.$$$ pro zápis (kde ufn=d:filename.typ).

Editaci je možno provádět postupně po stránkách (velikost je určena rozsahem volné paměti) ze vstupního souboru do pracovního.

Editaci lze ukončit dvojím způsobem – zrušením editace (Quit) či uzavřením editace (End). V případě zrušení editace uzavře ED vstupní soubor (zůstává beze změny – byl-li vytvářen nový soubor, není

zrušen a je prázdný) a zruší pracovní soubor. Při uzavření editace zkopíruje ED obsah paměti a zbytek vstupního souboru do pracovního souboru a dále zruší soubor d:filename.BAK (pokud existoval), přejmenuje původní vstupní soubor na d:filename.BAK (záložní kopie) a poté pracovní soubor na d:filename.typ.

Editovat je možno text uložený v operační paměti (viz přehled příkazů). Načtení (části) textu ze vstupu do paměti a zkopírování obsahu paměti na výstup je nutno řídit pomocí příkazů (Append, Write).

Rovněž lze text nebo jeho části kopírovat či číst z pomocných souborů – *knihovny* (musí být typu LIB). Opravovaný text je zobrazen pouze na požádání (příkaz Type) – ED není obrazovkový editor.

Příkazy pro program ED je možno zadávat pouze po výzvě ve tvaru " :* ". Před symbolem " : " může nebo nemusí být zobrazeno číslo aktuálního řádku (viz příkaz Visibility). Příkazy jsou jednoznakové, za některými příkazy mohou být uvedeny parametry (řetězy znaků). Omezovačem příkazů (parametrů) je konec řádku nebo znak "^Z". Má-li řetěz obsahovat konce řádku, musí být zadán jako znak "^L". Příkazy je možno zadávat jednotlivě nebo více najednou. Řádek se zadávaným příkazem je zpracováván až po ukončení (stisku RETURN či CR atd.). Před odesláním řádku v něm lze standardně editovat (viz editace příkazů). Většina příkazů se vztahuje k tzv. aktuální pozici (CP = Current Position).

Editační jednotkou může být znak, řetěz, řádek či skupina řádků. Některé příkazy je možno provádět vícekrát. Počet předepíšeme pomocí *opakovače*, kterým může být číslo v intervalu 0 až 65535 nebo znak "#", zastupující maximální možný počet opakování.

Chybové zprávy programu ED

? Neznámý příkaz.
> Operační paměť je zaplněna (nutno použít Write).
Příkaz není možno opakovat tolikrát, kolikrát bylo požadováno.
0 Nelze otevřít soubor typu LIB v příkazu Read.
E Zrušení činnosti příkazu (abort).
F Plný disk nebo adresář.

Přehled příkazů pro program ED

Příkazy uvádíme abecedně, n znamená opakovač (0...65535 nebo #). V kulatých závorkách je uvedena mnemotechnika (není součástí příkazu). Varianty příkazu jsou ve vysvětlení odděleny lomítkem (/). Symbol CP je ve vysvětlení použit jako zkratka pro výraz "aktuální pozice" (Ukazuje mezi znaky textu!). Hranaté závorky indikují volitelnou přítomnost části příkazu.

[n]A (Append) Připojí k textu v paměti jeden/n řádků ze vstupního souboru.

[-]B (Beginning/Bottom) Nastaví CP před začátek/za konec textu v paměti.

[-][n]C Posune CP o jeden/n znaků dopředu/zpět.

[-][n]D (Delete) Vymaže jeden/n znaků za/před CP.

E (End) Uzavření editace, předání řízení systému.

[n]Fstring (Find) Nalezení prvého/n-tého výskytu řetězu string. CP je umístěna za poslední znak řetězu. Není-li řetěz nalezen, CP se nemění.

H Pracovní uzavření a znovuotevření editace – přesun CP na začátek zpracovávaného souboru.

I (insert mode) Přechod do režimu vkládání, v němž lze vkládat libovolný text (s výjimkou speciálních znaků). Ve vstupním řádku lze editovat, i když není čten pomocí funkce 10 (BDOS). Speciální znaky "^H", "^X" a "^U" ruší text nejvýše do začátku aktuálního řádku. Znakem "DEL" lze rušit zpět i předcházející znaky textu. POZOR! Konec řádku v textu je tvořen znaky "CR" a "LF" ("LF" je automaticky doplněn při stisku "CR"). Rušíme-li konec řádku pomocí "DEL", musíme jej použít dvakrát! Není-li tato zásada dodržena, vznikne text, který neodpovídá zvyklostem CP/M. Z režimu vkládání se navrátíme do editace stiskem "^Z" a "CR" (RETURN).

Istring^Z (Insert string) Vložení řetězu string v místě CP.

InstringCR (Insert line) Vložení řetězu string v místě CP, ukončeného znaky "CR" a "LF" (konec řádku).

[n]Js1^Zs2^Zs3 (Juxtaposition) Najde řetěz s1, vloží za něj řetěz s2 a zruší všechny znaky až do prvního výskytu s3 (s3 zůstává), provede se n-krát.

[-][n]L (Line) Posune CP na začátek aktuálního řádku/n-tého řádku počínaje aktuálním směrem ke konci/k začátku textu.

[n]Mcmds (Multiple) Vykoná až do konce souboru/n-krát příkazy cmds.

[n]Nstring Hledání prvního/n-tého výskytu řetězu string v celém souboru (na rozdíl od F, který hledá pouze v paměti).

O (Origin) Vrátí editace do stavu při zahájení (návrat k původnímu souboru).

[-][n]P (Page) Posune CP o jednu stránku/n stránek směrem ke konci textu/k začátku textu a vždy zobrazí stránku (stránka je 23 řádků).

Q (Quit) Zrušení editace.

R[filename] (Read) Vloží soubor filename.LIB/X$$$$$$$.LIB v místě CP.

[n]Sold^Znew (Substitute) Nahradí první výskyt/n výskytů řetězu old řetězem new směrem ke konci textu.

[-][n]T (Type) Zobrazí jeden řádek/n řádků počínaje CP směrem ke konci/k začátku textu (neposouvá CP).

0T Zobrazí znaky od začátku aktuálního řádku do CP.

[-]V (Visibilty) Povolí/zakáže zobrazování čísel řádků.

0V Zobrazí údaje o velikosti vyrovnávací paměti a jejím obsazení.

[n]W (Write) Zapíše jeden řádek/n řádků uložených na počátku paměti do pracovního souboru a uvolní paměť.

[n]X Připojí jeden řádek/n řádků počínaje CP do pracovního souboru X$$$$$$$.LIB, aniž je ruší v paměti (viz Read).

0X Zruší pracovní soubor X$$$$$$$.LIB.

[n]Z (snooZe) Čeká n/2 sekund.

Speciální tvary příkazů

[-][n]=[-][n]LT Posune CP o jeden/n řádků směrem ke konci/začátku textu a zobrazí nastavený řádek.

x:cmds Před vykonáním příkazu(ů) cmds posune CP na řádek x:

:y cmds Vykoná cmds od CP do řádku y (včetně).

x: :y cmds Vykoná cmds od řádku x do řádku y (včetně).

Příklady použití kombinací příkazů

-B-TI	Vypíše poslední řádek v paměti a očekává vkládání textu (ukončen ^Z).
#M0LI;^Z0TL	Na začátek všech řádků v paměti vloží znak " ; " a zobrazí nový tvar textu.
20: :40M0LI(*^ZS^L^Z*)^L^Z0TL	Vloží na začátek a konec řádků 20 až 40 (včetně) komentářové závorky a současně zobrazuje výsledek (symbol ^L v řetězu symbolizuje konec řádku).
x: : yXBzLR	Přesune blok řádků x až y (včetně) za řádek z (nutno nejprve provést 0X).
BM-B-LXK2B#T0X	Obrátí pořadí řádků v paměti a vypíše je.

Poznámka: Dostupná verze programu ED obsahuje některé nepodstatné chyby:
– po provedení příkazu IstringCR se nápověda chybně zobrazí na začátku řádku;
– při pokusu o výpis textu před skutečným začátkem textu se poruší číslování řádků (někdy i při rušení řádků).

4.3.5 DUMP – Hexadecimální výpis obsahu souboru

DUMP ufn

Příkaz DUMP zobrazí na systémové konzoli šestnáctkově obsah souboru ufn ve tvaru:

```
0000 03 08 2D C3 F8 2C C3 B3 20 C3 19 2D 0B 00 0F 00
0010 18 00 00 00 00 00 00 20 20 43 4F 50 59 52 49 47
...
```

Číslování vlevo je rovněž šestnáctkové počínaje od nuly.

Poznámky: Tisk souboru lze zajistit stiskem ^P (viz speciální klávesy) před vyvoláním programu DUMP. Výpis na konzoli lze kdykoliv pozastavit stiskem ^S a opět pustit dále stiskem libovolné klávesy, tj. i třeba ^S nebo DEL nebo BACKSPACE (doporučujeme používat ^Q, kvůli kompatibilitě s jinými systémy).

4.3.6 ASM – Absolutní asembler pro 8080

ASM filename.parametry

Příkaz ASM slouží pro překlad zdrojového textu v jazyce symbolických adres do strojového kódu mikroprocesoru Intel 8080. Přívlastek absolutní vyjadřuje fakt, že výstupem je absolutní nepřemístitelný program, nikoliv přemístitelný modul, jak bývá u asemblerů zvykem. Firma Digital Research dodává asembler ASM spolu se služebními programy LOAD a DDT jako minimální prostředek pro vytváření programů.

Vstupní text je čten ze souboru filename.ASM. Asembler ASM vytváří soubor filename.HEX, obsahující strojový kód zapsaný v HEX formátu firmy Intel (viz přílohy) a soubor filename.PRN s protokolem o překladu. Vytvořený kód je nutno před spuštěním převést služebním programem LOAD do binárního tvaru. Umístění zdrojového textu a výstupních souborů lze řídit pomocí parameterů, které mají tvar:

.abc,

kde a je označení diskové jednotky se zdrojovým textem (A, ... , P);
b je označení diskové jednotky, kam má vystupovat přeložený program (A, ... , P nebo Z – potlačí generaci kódu);
c je označení diskové jednotky, kam má vystoupit protokol o překladu (A, ... , P nebo X – vypíše na konzoli nebo Z – potlačí výpis).
Překlad je prováděn běžným způsobem dvěma průchody – vytvářením tabulky symbolů a generováním kódu.

Tvar instrukce ASM

[číslo řádku#] [návěští [:]] operace [operandy] [;komentář]

Číslo řádku je nepovinné, ASM jej ignoruje. Jako návěští je možno použít libovolný identifikátor (musí začínat písmenem) o délce 1 až 16 znaků. Malá písmena v návěští jsou konvertována na velká. Uvnitř návěští se může vyskytovat znak "$", který ASM ignoruje. Za návěštím může a nemusí být uveden znak " : ". Jako návěští nelze použít rezervovaná slova – tj. symbolická označení instrukcí 8080, pseudoinstrukce (direk-

tivy – viz dále) a označení registrů (A, B, C, D, E, H, L, M, SP, PSW). Na místě operace lze použít libovolný symbol instrukce pro 8080 nebo pseudoinstrukci ASM (direktivu) nebo instrukci pro 8080. Pro pseudoinstrukce ASM (ORG, END, EQU, SET, IF, DB, DW, DS) a operandy platí zásady uvedené v odst. 2.3.3.

Chyby hlášené ASM během překladu:

NO SOURCE FILE PRESENT	Nenalezne zdrojový text na udané deskové jednotce.
NO DIRECTORY SPACE	Není možno vytvořit výstupní soubory – diskový adresář je plný.
SOURCE FILE NAME ERROR	Chybné jméno.
SOURCE FILE READ ERROR	Chyba při čtení.
OUTPUT FILE WRITE ERROR	Chyba při zápisu (pravděpodobně plný disk).
CANNOT CLOSE FILE	Chyba při uzavření (pravděpodobně je disk chráněn proti zápisu).

V protokole o překladu (PRN):

D (Data error)	Data nelze umístit v udané oblasti.
E (Expression error)	Chybně vytvořený výraz, jeho hodnotu nelze určit v době překladu.
L (Label error)	V daném kontextu nelze použít toto návěští (např. dvojí deklarace).
N (Not implemented)	Daná verze ASM neumí přeložit.
O (Owerflow)	Příliš složitý výraz.
P (Phase error)	Návěští nemá stejnou hodnotu v obou fázích překladu (např. chybné SET).
R (Register error)	Chybné označení registru.
V (Value error)	Chybný operand ve výrazu.

4.3.7 LOAD – Převod z tvaru HEX na COM

LOAD ufn

Program LOAD převádí výstup asembleru ASM (soubor typu HEX – viz přílohy) na zaveditelný binární soubor typu COM. Soubor typu HEX obsahuje znakově zapsaný, šestnáctkově vyjádřený binární kód.

Poznámky: Typ vstupního souboru musí být HEX a není nutno jej uvádět (LOAD A.HEX je totéž jako LOAD A).

Výstupní soubor má stejné jméno jako vstupní soubor, je typu COM a je umístěn na stejném disku jako vstupní soubor.

Vstupní soubor musí obsahovat platný program ve tvaru HEX (dle INTEL), který bude umístěn od adresy 100H. Adresy v záznamech HEX musí být ve správném pořadí, prázdná místa jsou vyplněna nulami. Programy umístěné jiným způsobem musí být převedeny na formát COM pomocí služebního programu DDT.

4.3.8 DDT – Ladicí prostředek pro procesor 8080

DDT [_ ufn]

Program DDT (Dynamic Debugging Tool) je určen pro přímé interaktivní ladění programů ve strojovém kódu/asembleru 8080. Pokud je v příkazu uveden parametr ufn, je odpovídající program zaveden do paměti (viz příkazy Input a Read). Během ladění je možno zobrazovat a měnit obsah paměti, pokusně provádět úseky programu apod. Odladěný program je možno uložit pomocí zabudovaného příkazu SAVE, jak to demostruje např. následující protokol:

```
A>DDT TEST.COM      ..... spuštění programu DDT
DDT VERS 2.2
NEXT PC             ..... informace o délce a startovací adrese
0400 0100                 zavedeného programu TEST
-L100,100           ..... výpis instrukce na adrese 100H
0100 LXI H,0003
-A100               ..... změna programu od adresy 100H
0100 NOP
```

```
0101 NOP
0102 NOP
0103 <cr>                  ..... ukončení změn programu
-GO                        ..... skok na adresu 0 (ukončení DDT)
A>SAVE 3 TEST.COM          ..... uložení opraveného programu
A>
```

Poznámka: Program DDT je standardně zaveden procesorem příkazů CCP do operační paměti od adresy 100H. Po spuštění se program DDT přemístí v paměti na místo procesoru příkazů CCP a modifikuje obsah adres 6 a 7 v komunikační zóně tak, aby přesměroval všechna volání služeb systému do sebe. Tím současně nastaví konec volné paměti pod sebe, aby nebyl přepsán programy, které využívají celou dostupnou paměť (zjišťují velikost volné paměti z obsahu adres 6 a 7). Program DDT dále uloží na adresu 38H instrukci skoku na svůj vstupní bod pro pracovní pozastavení programu (breakpoint), které lze pak realizovat instrukcí RST 7. Instrukci RST 7 umísťuje program DDT do místa pro pracovní zastavení programu. Výše uvedený postup zajistí uvolnění zóny TPA pro laděný program.

Přehled příkazů DDT

Příkazy je možno zadávat po zobrazení výzvy " – " v prvé pozici.

A (Assembler)	Vstup instrukcí v asembleru.
D (Dump)	Výpis obsahu paměti šestnáctkově a znakově.
F (Fill)	Naplnění bloku paměti konstantou.
G (Go)	Zahájení provádění (části) programu.
H (Hex)	Šestnáctková aritmetika (+, -).
I (Input)	Nastaví vstup pro příkaz R.
L (List)	Výpis obsahu paměti ve formě instrukcí v asembleru.
M (Move)	Přesun bloku paměti.
R (Read)	Čtení souboru nastaveného příkazem I do paměti.
S (Substitute)	Opravy obsahu paměti.
T (Trase)	Trasování po instrukcích.
U (Untrace)	Trasování po blocích.
X (eXamine)	Výpis a modifikace obsahu registrů.

Popis příkazů pro program DDT

Příkazy pro program DDT jsou zadávány z klávesnice po výzvě " - ". Parametry příkazů jsou poziční a jsou zadávány jako hexadecimální čísla (bez uvádění příznaku H). Pro aktuální pozici v programu (čítač adres) je ve vysvětlení použita zkratka PC (Program Counter). Při zadávání příkazu pro program DDT je možno používat standardní způsob oprav příkazů v systému CP/M (příkazy jsou čteny systémovou službou 10 modulu BDOS). Příkaz může mít maximálně 32 znaků. Neznámý příkaz, chybu v parametrech, či chybu při provádění příkazu ohlásí program DDT výpisem znaku "?".

A[adresa]

Vstup instrukcí v asembleru a přímé obsazování paměti překladem symbolických instrukcí od aktuálního čítače adres PC nebo od určené adresy. Vstup je ukončen zasláním prázdné řádky (stiskem RETURN, CR apod.). Čítač adres PC se posouvá současně s ukládáním instrukcí a je zobrazován šestnáctkově jako výzva pro zadání další instrukce.

D[adresa1 [,adresa2]]

Zobrazí na systémové konzoli obsah paměti od aktuální hodnoty čítače adres PC (resp. od adresy naposledy zobrazené příkazem D, resp. zadané adresy adresa1) v délce 256 slabik (resp. až po zadanou adresu adresa2) ve tvaru:

```
-D100,120
0100 C3 08 2D C3  F8 2C  C3 B3  20 C3  19 2D 0B  00 0F 00 . . - . . , .
0110 18 00 00  00  00 00  00 20  20 43  4F 50  59  52 49 47 . . . . . .
0120 48
```

(po pravé straně je uveden znakový výpis v kódu ASCII, nezobrazitelné znaky jsou nahrazeny znakem " . ")

Fadresa1, adresa2, znak

Vyplní operační paměť od adresy adresa1 po adresu adresa2 (včetně) udaným znakem.

G[adresa1] [,adresa2 [,adresa3]]

Předá řízení programu na aktuální hodnotu čítače adres PC (resp. udané adresy adresa1). Na adresy adresa2 (a příp. adresa3) nastaví instrukce RST 7, která předá řízení zpět programu DDT. Po návratu se nastavené body přerušení opět obnoví původním obsahem.

H, a, b

Zobrazí šestnáctkově hodnoty a + b a a - b.

Ifilename [.typ]

Nastaví jméno souboru (příp. i typ souboru) do implicitního popisu souboru v komunikační zóně (adresa 5CH). Tento popis využívá příkaz Read, nebo jej může používat laděný program.

L[adresa1[,adresa2]]

Zobrazí obsah paměti od aktuální hodnoty čítače adres PC (resp. od adresy naposledy vypsané příkazem L, resp. od udané počáteční adresy adresa1) ve formě zpětného překladu do asembleru 8080. Zobrazí se buď 12 instrukcí, nebo výpis pokračuje až do uvedené koncové adresy adresa2.

Madresa1,adresa2,adresa3

Zkopíruje blok paměti od počáteční adresy adresa1 do koncové adresy adresa2 (včetně) na novou adresu adresa3. Přesun se provádí po slabikách, počínaje od nižších adres.

R[báze]

Zavede do paměti soubor, jehož popis je v komunikační zóně (adresa 5CH) (viz příkaz Input). Soubor typu HEX se zavádí od ukládací adresy (příp. ukládací adresy + báze), soubory ostatních typů se zavádějí od adresy 0100H (příp. 0100H + báze). Takto lze zavádět programy i na jinou adresu než 100H.

Sadresa

Zobrazí adresu a hexadecimální obsah paměti na uvedené adrese jako výzvu k zadání nového obsahu. Stiskem znaku různého od hexadecimální číslice nebo RETURN, se ukončí náhrada obsahu paměti, stisk samotného RETURN způsobí přechod na další adresu beze změny obsahu. Rovněž po udání nového obsahu se přejde na další adresu.

T[počet]

Vypíše obsah všech registrů procesoru (jako příkaz X) a provede instrukci určenou aktuálním obsahem čítače adres PC. Pokud je udán počet opakování, opakuje se dle udaného počtu.

U[počet]

Totéž jako příkaz Trace, ale zobrazení se neprovede po provedení každé instrukce, ale pouze na konci.

X[{registr nebo příznak}]

Zobrazí obsah registrů procesoru ve tvaru:

CxZxMxExIx A=xx B=xxxx D=xxxx H=xxxx S=xxxx P=xxxx instrukce

(instrukce je instrukce na adrese v čítači adres PC, který je zobrazen jako P). Uvedeme-li v příkazu X jako parametr registr nebo příznak, zobrazí se pouze obsah registru či příznaku jako výzva pro zadání nové hodnoty. Pouhý stisk RETURN ponechá hodnotu nezměněnu, jinak se obsah nastaví dle nové hodnoty.

4.3.9 SYSGEN – Kopírování a ukládání CP/M

SYSGEN

Program SYSGEN slouží ke kopírování, resp. k ukládání obrazu operačního systému CP/M na standardní osmipalcové diskety. Příkaz pro vyvolání programu SYSGEN má tvar:

A>SYSGEN

Program se ohlásí

SYSGEN VERSION 2.x
SOURCE DRIVE NAME (OR RETURN TO SKIP) .

V tomto okamžiku záleží na tom, jedná-li se o kopírování systému nebo jeho uložení. Je-li operační sytém uložen v paměti od adresy 900H (900-97FH systémový zavaděč, 980H-1180H modul CCP, 1180H-1F7FH modul BDOS, 1F80H modul BIOS), stiskne se klávesa RETURN. V opačném případě se zadá jednotka, ze které se má (z prvních dvou stop) obraz systému do paměti na udané adresy přečíst. Na zadání jednotky reaguje SYSGEN zprávou:

SOURCE ON n:, THEN TYPE RETURN.

Po založení diskety a stisku klávesy je systém načten a vypíše se zpráva:

DESTINATION DRIVE NAME (OR RETURN TO SKIP)

Je třeba odpovědět označením jednotky, na které budeme nahrávat (zde symbolicky n). Program SYSGEN reaguje zprávou:

DESTINATION ON n, THEN TYPE RETURN

Do zvolené jednotky se založí disketa a po stisku klávesy se nahraje. Poté se SYSGEN opět ohlásí

FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)

a nahrávání se může opakovat nebo ukončit.

4.3.10 MOVCPM – Rekonfigurace CP/M

MOVCPM[_{n/*}[_*]]

Program MOVCPM je určen pro přeadresaci CP/M pro různé velikosti operační paměti – rezidentní část systému je umístěna na nejvyšších adresách. Systém CP/M umí ovládat paměť pouze do svého počátku (pod sebou). Vyvolání programu MOVCPM přeadresuje rezidentní část CP/M tak, aby uměla pracovat v paměti, jejíž velikost udává první parametr (n = velikost paměti v KB, * znamená maximální možnou na dané konfiguraci).

Příklady použití MOVCPM

MOVCPM — Program MOVCPM zavede systém ze systémové oblasti disku do paměti a spustí.

MOVCPM 48 — Program MOVCPM zavede systém ze systémové oblasti na disku a přeadresuje jej tak, aby uměl pracovat na konfiguraci s pamětí 48 KB. Po skončení programu MOVCPM je nutno použít služební program SYSGEN pro uložení obrazu nového systému na disk.

MOVCPM * * — Program MOVCPM zavede systém ze systémové oblasti na disku a přeadresuje jej tak, aby byla maximálně využita paměť stávající konfigurace. Poté vyzve k uložení systému zprávou: SAVE 32 CPMxx.COM.

5 Programové prostředí systému CP/M

Programové prostředí systému CP/M vytváří jádro systému. Jádro je umístěno rezidentně v operační paměti a poskytuje ostatním programům služby při spolupráci s technickými prostředky. Jádro operačního systému CP/M tvoří moduly BDOS a BIOS (kap. 3.2). V této kapitole se budeme zabývat službami modulu BDOS. Jsou zde uvedeny informace potřebné při vytváření programů, které pracují ve standardním programovém prostředí systému CP/M. Modul BIOS je popsán v kap. 6.

5.1 Rozdělení operační paměti

Správa operační paměti systému CP/M vychází z předpokladu, že v technických prostředcích hostitelského mikropočítače neexistují žádné prostředky pro realizaci ochrany paměti. Pro monoprogramový a monouživatelský systém z toho plyne nemožnost ochrany systému proti zásahům ze strany aplikačních programů. Pro zachování jednoduché modifikovatelnosti a přenosnosti systému CP/M není dokonce ani využita možnost použití pevných pamětí pro uložení kódu systémových programů. Není s ní ani počítáno při jejich tvorbě, uložení kódu systému do pevných pamětí vyžaduje modifikace systému, které nejsou nijak podporovány.

Správa operační paměti se proto redukuje na prosté rozdělení operační paměti na úseky a přidělení jednotlivých úseků paměti komponentám programového vybavení. Jednotlivé části systému jsou (obvykle) umístěny v paměti podle schématu na obr. 3.1. Označení v obrázku mají následující význam.

Označení	Délka	Význam
SPA	100H	komunikační zóna (System Parameter Area)
TPA	různá	zóna transientních programů (Transient Program Area)
CCP	2 KB	procesor řídicích příkazů (Console Command Processor)
BDOS	3,5 KB	základní diskový operační systém (Basic Disk Operating System)
BIOS	různá	základní systém vstupu a výstupu (Basic Input/Output System)

Jak je z obrázku patrno, je dostupná operační paměť rozdělena na tři základní úseky:
- rezidentní oblast systému (BDOS a BIOS),
- komunikační zónu mezi systémem a programy (SPA) a
- oblast programů (TPA), kterou využívá i nerezidentní část systému – procesor příkazů CCP.

Rezidentní oblast systému je umístěna na konec operační paměti, protože modul BIOS je pro každou instalaci nutno modifikovat podle konkrétního technického prostředí. Tím se mění jeho velikost a s ní i rozsah použité rezidentní oblasti. Pokud by byla umístěna na počátku, měnilo by se pro různé instalace systému umístění oblasti TPA. Instrukční repertoár procesorů typu 8080 nedovoluje vytvářet efektivní kód, který je možno provozovat v libovolné oblasti paměti. Změna počátku oblasti TPA by tak měla za následek nutnost přeadresování všech nerezidentních programů.

Počátek rezidentní oblasti je určen velikostí rezidentního kódu systému a skutečným rozsahem operační paměti. Může se proto opět pro různé instalace sytému lišit. Proto je pro komunikaci mezi programy a systémem vyhrazena komunikační zóna SPA, která je umístěna pevně na počátku operační paměti (obvykle od adresy 0). Tato koncepce dovoluje např. standardní volání služeb systému přes pevnou adresu v komunikační zóně, bez ohledu na velikost použité operační paměti.

Pevně je rovněž stanoven počátek uživatelsky použitelné oblasti paměti (TPA), kde se střídají služební a aplikační programy – označujeme je proto shodně jako transientní (pomíjivé) programy.

Umístění procesoru příkazů CCP vyplývá z jeho poslání. Procesor CCP je program, mezi jehož funkce patří i zavádění ostatních transientních programů, se kterými se v paměti střídá. Zavedený program nemusí přepsat oblast paměti obsazenou CCP – v tom případě není nutno po skončení činnosti programu v TPA znovu zavádět procesor příkazů CCP z disku.

Skutečná velikost operační paměti, ve které provozujeme operační systém CP/M, se může měnit v rozsahu od 20 KB do 64 KB (verze 1.x lze provozovat i v paměti o rozsahu od 16 KB, verze 3.x a vyšší lze upravit i pro paměť nad 64 KB). Originální systém je distribuován ve verzi pro 20 KB operační paměti. Pomocí služebních programů SYSGEN a MOVCPM lze rekonfigurovat (při generování systému) moduly CCP a BDOS pro umístění od libovolné adresy dělitelné 256. Podrobně se generováním systému zabývá kap. 6. V následujících odstavcích nejprve probereme obsah komunikační zóny a poté se věnujeme využívání programových služeb jádra CP/M.

5.1.1 Komunikační zóna SPA

Komunikační zóna SPA slouží pro předávání parametrů mezi operačním systémem a programem běžícím v zóně TPA. V tomto odstavci podrobně probereme obsah komunikační zóny a význam jednotlivých položek. Na položky komunikační zóny se budeme odkazovat přímo pomocí adresy.

Adresa	Délka	Význam
00H	3	instrukce skoku na službu modulu BIOS, která provede teplý start systému (JP WBOOT),
03H	1	aktuální přiřazení logických a fyzických zařízení (IOBYTE),
04H	1	aktuální uživatel (bity 7 až 4) a aktuální disková jednotka (bity 3 až 0) (CDISK),
05H	3	instrukce skoku na začátek modulu BDOS (JP BDOS+6),
08H	40	rezervováno pro zpracování přerušení 1 až 5 (RST1 až RST5); tato oblast není systémem využívána,
30H	8	přerušení 6 (RST6); rezervováno pro systém,

38H	3	přerušení 7 (RST7); tato instrukce je využívána ladícími programy DDT, SID, ZSID pro trasování laděných programů; vlastním systémem není využívána,
3BH	5	rezervováno pro systém,
40H	16	rezervováno pro systém,
50H	12	rezervováno pro systém,
5CH	36	implicitní řídicí blok (FCB1),
6CH	20	druhý neúplný implicitní řídicí blok (FCB2)
80H	128	vyrovnávací paměť (implicitní zóna DMA).

Adresa 00H: Volání inicializace systému

Na adrese 0 je umístěna instrukce skoku na službu modulu BIOS (JP WBOOT), která se nazývá teplý start systému. Skokem na adresu 0 musí nutně končit aplikační programy, které využívají paměť obsazenou moduly CCP a BDOS. Pokud program nepřepisuje oblast modulu BDOS, lze pro návrat do systému použít i službu 0 modulu BDOS. Neporuší-li program ani oblast obsazenou modulem CCP, lze jej ukončit přímým návratem do CCP. Obvykle však programy končí skokem na adresu 0, neboť teplý start zaručí bezchybnou funkci modulů CCP a BDOS.

Instrukce skoku na adrese 0 směřuje na druhý prvek ve skokovém vektoru modulu BIOS. Adresní část instrukce tedy udává počáteční adresu modulu BIOS zvětšenou o tři (BIOS+3). Této skutečnosti lze využít při vytváření programů, které přímo komunikují s modulem BIOS a přitom nechtějí být závislé na konkrétní konfiguraci operačního systému (jeho umístění v paměti). Instrukce volání jednotlivých funkcí modulu BIOS se v takovémto programu musí modifikovat podle obsahu adresy 2.

Obsah adresy 2 lze využít pro zjištění začátku modulu BIOS. Modul BIOS je vždy v paměti uložen tak, aby začínal na adrese dělitelné 256 (od počátku stránky). Instrukce skoku je uložena v paměti ve třech slabikách: operační znak, nižší bity adresy, vyšší bity adresy. Proto je na adrese 2 uložena stránka, od které je uložen modul BIOS. Obsah komunikační zóny od adresy 0 lze zobrazit pomocí služebního programu DDT, jak to demonstruje následující ukázka.

```
A>DDT          ....    vyvolání programu DDT
-d0,2          ....    zobrazení obsahu paměti
0000: C3 03 F2         na adresách 0, 1 a 2
-l0, 2         ....    zobrazení instrukce, která
0000: JMP F203         je uložena na adrese 0
0003:
-g0            ....    návrat z DDT
A>
```

Počátek modulu BIOS lze zjistit např. následujícím programem.

```
BIOSP EQU 2
      LD  A,(BIOSP)
; zde registr A obsahuje stránku, ve které začíná BIOS
```

Adresa 03H: Přiřazení fyzických zařízení logickým

Na adrese 03H je umístěna standardní slabika označovaná IOBYTE (dle značení firmy INTEL), která určuje aktuální přiřazení fyzických periferních zařízení zařízením logickým. Každá dvojice bitů slabiky IOBYTE koresponduje s jedním logickým zařízením a jejich čtyři možné kombinace udávají aktuálně přiřazené fyzické zařízení. Podrobně je obsah slabiky IOBYTE popsán v tab. 6.3 v kap. 6. Slabiku IOBYTE lze číst a nastavit pomocí služeb modulu BDOS. Nejsou-li v některých variantách tyto služby k dispozici, nelze využívat všechna fyzická periferní zařízení programem PIP. Vlastním jádrem operačního systému není slabika IOBYTE využívána.

Adresa 04H: Aktuální uživatel a disková jednotka

Slabika na adrese 4 obsahuje informaci o aktuálním uživateli a aktuální diskové jednotce. Spodní čtyři bity slabiky na adrese 04H (CDISK) udávají číslo aktuální diskové jednotky (0 = A, ... , 15 = P). Horní čtyři bity udávají číslo aktuálního uživatele (0 až 15).

Adresa 05H: Volací adresa služeb modulu BDOS

Adresa 5 v zóně SPA slouží jako volací adresa služeb modulu BDOS. Je na ní umístěna instrukce skoku na počátek modulu BDOS (JP BDOS+6) Využívá se pro dva účely. Instrukcí CALL 5 se volají služby modulu BDOS. To znamená, že o služby modulu BDOS se žádá vždy stejně, bez ohledu na umístění operačního systému v paměti. Adresní část instrukce skoku na adrese 5 pak udává konec operační paměti přístupné uživateli (neboť modul CCP není při běhu programu využíván a může být proto přepsán).

Poznámka: Program DDT se zavádí do paměti místo modulu CCP těsně pod rezidentní oblast systému, a proto modifikuje obsah adresy 5 tak, aby nebyl přepsán programy, využívajícími dynamicky celou dostupnou paměť. Pro zobrazení obsahu adresy 5 nelze proto použít program DDT, neboť obsah adresy 5 udává při zobrazení pomocí DDT počátek programu DDT.

Následující program uloží do registrového páru HL poslední použitelnou adresu před rezidentní zónou systému.

```
TOP:  EQU 7
      LD  A,(TOP)   ; stránka, ve které začíná BDOS
      DEC A         ; předchozí stránka
      LD  L,0FFH    ; poslední slabika v této stránce
      LD  H,A       ; HL ukazuje na poslední volnou adresu
```

Adresa 5CH: Implicitní řídicí blok

Na adrese 5CH je umístěn systémem vytvořený řídicí blok souboru (FCB1). Jsou-li v příkazové řádce přečtené procesorem příkazů CCP za jménem příkazu uvedeny parametry, jsou považovány (až po mezeru) za identifikaci souboru. Tato identifikace (může být i nejednoznačná) je modulem CCP dekódována a převedena do implicitního řídicího bloku FCB1 na adrese 5CH. Je-li v nejednoznačné identifikaci uvedena hvězdička, je nahrazena znaky '?' až do konce příslušné položky. Takto inicializovaný řídicí blok může program přímo používat pro práci se souborem.

Následuje-li po prvém parametru v příkazu za mezerou další text, provede se opět dekódování a nastaví se druhý implicitní řídicí blok (FCB2) začínající na adrese 6CH. Tento řídicí blok si musí program přesunout do vlastní oblasti paměti dříve než začne pracovat s blokem na adrese 5CH, neboť pak se obsah paměti na adresách 6CH až 7FH přepíše. Oba bloky se totiž částečně překrývají.

Používá-li se při zpracování souboru s řídicím blokem na adrese 5CH přímý přístup k větám souboru, je číslo věty (r0, r1, r2) pro přímý přístup uvedeno na adresách 7DH až 7FH.

Adresa 80H: Vyrovnávací paměť

Zóna paměti na adresách 80H až 0FFH je využívána jako implicitní vyrovnávací paměť pro diskové vstupně/výstupní operace (adresa DMA). Současně slouží pro předávání parametrů uvedených v příkazu ke spuštění programu. Při spuštění programu je vyrovnávací paměť naplněna zbytkem příkazové řádky za jménem programu, přičemž na adrese 80H je v jedné slabice uložena délka tohoto zbytku. Vlastní text začíná na adrese 81H oddělovací mezerou.

5.2 Struktura programu

Programy, které mají pracovat v prostředí systému CP/M a které lze zavádět a spouštět pomocí procesoru příkazů CCP, musí být vytvořeny podle standardních pravidel.

1. Každý program, který spouští procesor příkazů CCP, musí být uložen v souboru typu COM.

2. Každý program musí být organizován tak, aby začínal na adrese 100H (např. pseudoinstrukcí ORG 100H).

3. Startovací adresa programu musí být adresa 100H. V minimálním případě musí tedy program začínat instrukcí JP START.

4. Program může využívat operační paměť v rozsahu oblasti TPA, tj. do začátku paměti, rezidentně obsazené systémem. Rozsah oblasti TPA lze zajistit z obsahu komunikační zóny SPA např. programem, který nastaví ukazatel zásobníku na poslední volnou adresu.

```
TOP:  EQU  7
      LD   A,(TOP)     ; stránka, ve které začíná BDOS
      DEC  A           ; předchozí stránka
      LD   L,0FFH      ; poslední slabika v této stránce
      LD   H,A         ; HL ukazuje na poslední volnou adresu
      LD   SP,HL       ; nastav ukazatel zásobníku
```

5. Program by měl skončit standardním návratem do systému, tj. buď voláním služby 0 modulu BDOS

```
LD   C,0
CALL 5
```

nebo teplým startem systému (tj. instrukcí JP 0). Pokud program nepřepisuje rezidentní paměť systému, postačí zcela volání služby 0 (ušetří se zavádění modulu BDOS); pokud si nejsme jisti, je lépe použít teplý start.

6. Pokud program nepřepisuje ani paměť, v níž je uložen procesor příkazů CCP, lze jej ukončit instrukcí RET, neboť procesor příkazů CCP spouští každý program instrukcí CALL 100H. Ušetří se tak i zavádění procesoru příkazů CCP. V okamžiku provedení této instrukce je ukazatel zásobníku nastaven do zóny v procesoru příkazů, která má délku 16 slabik. Využívá-li program zásobník do větší hloubky (více než 7 adres), je nutno přepnout ukazatel zásobníku do vlastní paměti.

```
        ORG    100H
START:
        LD     HL,0              ; úschova SP
        ADD    HL,SP
        LD     (CCPSTACK),HL
        LD     SP,NEWSTACK       ; nový ukazatel zásobníku
        ...                      ; tělo programu
        LD     HL,(CCPSTACK); obnova SP
        LD     SP,HL
        RET                      ; návrat do CCP

CCPSTACK:  DS 2
           DS ...                ; místo na nový zásobník
NEWSTACK   EQU *
```

7. Pro spolupráci s prostředím by měl program zásadně používat služeb modulu BDOS, popsaných v následujících odstavcích. Při dodržení tohoto pravidla lze zaručit, že program bude pracovat stejně na libovolné instalaci systému CP/M, pokud zde budou dostatečné technické prostředky (paměť, přídavná zařízení atd.).

5.3. Systém ovládání souborů

Jedna z hlavních funkcí modulu BDOS je zajištění logické organizace dat na diskových pamětech (obecněji na vnějších pamětech s přímým přístupem). V tomto odstavci je podrobně popsán vztah mezi logickou organizací (soubory, věty) a fyzickou organizací (stopy, sektory) dat na diskových médiích.

5.3.1 Logická organizace souborů

Z hlediska transientních programů (uživatelů služeb modulu BDOS) je vnější paměť tvořena nejméně jednou až maximálně šestnácti logickými diskovými jednotkami. Logické diskové jednotky jsou v rámci systému identifikovány písmeny A až P.

Do každé logické diskové jednotky může být umístěno médium, které je strukturováno do informačních celků nazývaných soubory. Vzhledem k dynamickému charakteru existence souborů, které vznikají a zanikají, jsou soubory identifikovány pomocí jmen.

Soubory je možno vytvářet, rušit, vyhledávat podle identifikace, zapisovat do nich nebo z nich číst. Každý soubor se skládá z logických vět pevné délky 128 slabik, které představují informační jednotku pro komunikaci při čtení nebo zápisu. Logické věty souboru jsou identifikovány pořadím v rámci souboru, nazývaným číslo věty. K větám souboru je možno přistupovat sekvenčně nebo přímým přístupem podle čísla věty. Maximální teoretická velikost jednoho souboru je 65 536 logických vět (tj. 65 536 × 128B = 8MB).

Soubor uložený na vnější paměti je identifikován trojicí

[jednotka:] jméno [.typ] ,

kde: jednotka je označení diskové jednotky (písmena A až P u verze 2.x a verzí vyšších, A až D u verze 1.4 a verzí nižších),
jméno je jméno souboru (maximálně 8 znaků),
typ je typ souboru (maximálně 3 znaky)

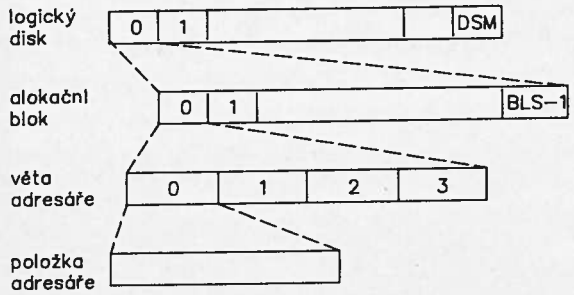
Poznámka: U verze 3.1 (CP/M PLUS) je dále možno ke každému souboru přiřadit heslo chránící přístup k souboru, heslo je maximálně osmiznakové a uvádí se na konci označení souboru oddělené znakem " ; ".

5.3.2 Fyzická struktura vnějších médií

Předpokládaná fyzická struktura záznamu na vnějších pamětech s přímým přístupem pracuje se sektory pevné délky. Záznam se provádí ve stopách, přičemž všechny stopy obsahují stejný počet sektorů. Stopy se číslují od vnějšího okraje média, počínaje od 0, až po počet stop mínus 1. Sektory na stopě se číslují od 1 až po počet sektorů na stopě (SPT – Sector Per Track). Má-li záznamové médium více aktivních povrchů, je navíc třeba rozlišovat číslo povrchu (hlavy).

Původně systém CP/M umožňoval práci pouze se standardními 8palcovými jednotkami pružných disků ve formátu IBM 3740 (77 stop po 26 sektorech délky 128 slabik). Od verze 2.0 je možno využívat prakticky libovolnou vnější paměť s přímým přístupem. Charakteristiky vnější paměti se udávají v tabulkách modulu BIOS, které jsou přístupné pomocí služeb modulu BDOS.

Informace potřebné k transformaci mezi logickou a fyzickou strukturou jsou rovněž uloženy na médiu ve formě tzv. *adresáře média.* Transformace mezi logickou a fyzickou organizací probíhá ve dvou stupních, přes tzv. *logickou diskovou jednotku.* V hlavičce souboru (v adresáři) je popsáno



Obr. 5.1. Struktura logické diskové jednotky

umístění dat v rámci logické jednotky. Logická jednotka je potom mapována na jednotku fyzickou.

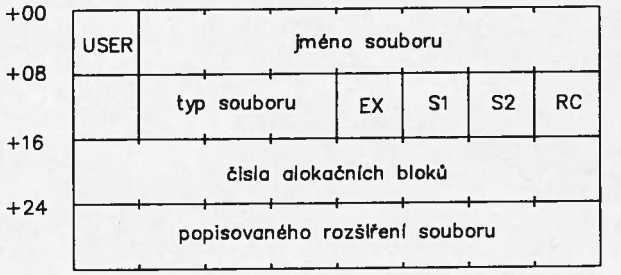
Logická disková jednotka (viz obr. 5.1) je tvořena logickými větami pevné délky 128 slabik. Maximální teoretický počet logických vět v souboru je 65 536 (0. až 65 535. věta). Logické věty jsou rozděleny do tzv. *alokačních bloků*. Velikost alokačního bloku, tj. počet logických vět tvořících alokační blok, se označuje jako BLS (Block Size) a je to vždy mocnina 2, větší nebo rovna 8 (tj. 8, 16, ... , max 128). Alokační bloky jsou číslovány počínaje od 0 až po konstantu označenou BLM (Block Mask). Skutečná kapacita jednotky je tedy (BLM+1) × BLS vět.

5.3.3 Adresář disku

Alokační bloky logické diskové jednotky jsou rozděleny na datové bloky a bloky adresáře. Datové alokační bloky obsahují datové věty jednotlivých souborů. Struktura a obsah datových vět nejsou z hlediska systému definovány.

Alokační bloky adresáře jsou bloky s nejnižšími čísly. Kromě bloku s číslem 0, který je vždy adresářový, to mohou být i bloky vyšších čísel (maximálně 16 bloků, tj. 0. až 15. alokační blok).

Každá věta adresáře obsahuje čtyři položky adresáře. Položky adresáře mají pevnou délku 32 slabik. Umístění položky v rámci logické věty (0, 1, 2 nebo 3) se nazývá *adresářový kód*. Položka adresáře popisuje jeden soubor nebo jeho část. To znamená, že zajišťuje vazbu mezi jménem souboru a fyzickým místem uložení jeho obsahu, příp. části obsahu.



Obr. 5.2. Struktura položky adresáře

Struktura položky adresáře:

Název	Délka	Význam
user	1	číslo uživatele (User Code)
f1 až 8	8	jméno souboru (File Name)
t1 až 3	3	typ souboru (File Type)
ex	1	číslo rozšíření (Extent)
s1	1	rezervováno
s2	1	vyšší bity čísla rozšíření
rc	1	počet vět v rozšíření (Record Count)
d0 ... d15	16	čísla alokačních bloků rozšíření

Strukturu řídicího bloku souboru lze rovněž vyjádřit následující deklarací v jazyce Pascal.

```
type
  BYTE        = 0..255;
  FILENAME    = packed array [1..8] of char;
  FILETYPE    = packed array [1..3] of char;
  TYPADRITEM  = packed record {položka adresáře}
                USER :BYTE ;      {uživatel (0-15)}
                FN   :FILENAME ;  {jméno souboru}
                FT   :FILETYPE ;  {typ souboru a atributy}
                EX   :BYTE ;      {číslo akt. rozšíření (0-31)}
                S1   :BYTE ;      {využívá BDOS}
                S2   :BYTE ;      {využívá BDOS}
                RC   :BYTE ;      {počet vět v rozšíření}
                DADR : packed array [1 . . 16] of char ; {čísla
                                    alokačních bloků daného
                                    rozšíření}
              end;
```

V jazyce symbolických adres používáme následující deklarace pro vyjádření struktury položky adresáře.

```
; POLOŽKY ADRESÁŘE

ADRUSER   EQU ADR ; OZNAČENÍ DISK. JEDNOTKY
ADRNAME   EQU ADR+01 ; JMÉNO SOUBORU
ADRTYPE   EQU ADR+09 ; TYP SOUBORU
ADREX     EQU ADR+12 ; ČÍSLO AKTUÁL. ROZŠÍŘENÍ
ADRS1     EQU ADR+13
ADRS2     EQU ADR+14
ADRRC     EQU ADR+15 ; POČET VĚT
ADRDADR   EQU ADR+16 ; ČÍSLA ALOKAČNÍCH BLOKŮ
```

Význam prvků položky adresáře:

user – číslo uživatele, kterému soubor patří

Povolená hodnota je 0 až 15 nebo 0E5H. Kombinace 0E5H je použita jako příznak, že položka adresáře je volná.

Poznámka: Standardně předznačený disk je popsán slabikami 0E5H a je z hlediska systému CP/M prázdný. Disk používaný jiným systémem musí být před použitím v systému CP/M znovu předznačen nebo alespoň popsán v adresářové části slabikami 0E5H, jinak systém CP/M pracuje chybně.

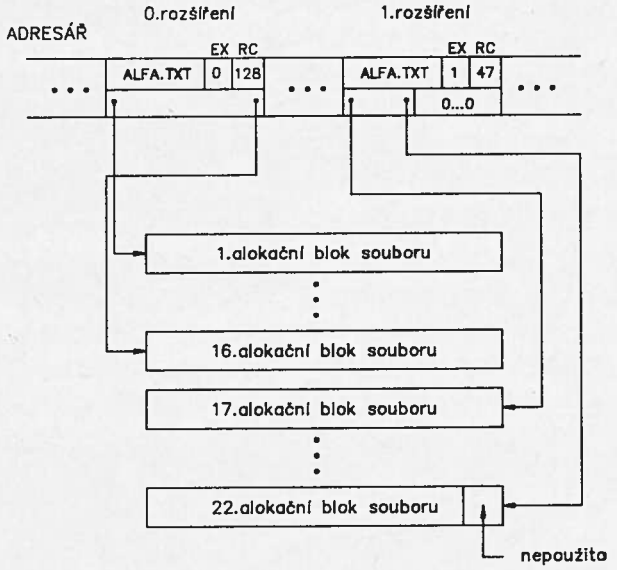
f1 až f8 – jméno souboru
t1 až t3 – typ souboru

Jméno i typ souboru jsou uvedeny znakově v sedmibitovém kódu ASCII. Pokud má jméno méně než 8 znaků, je doplněno mezerami. Podobně typ je případně doplněn mezerami na 3 znaky. Jménem a typem je soubor identifikován při vyhledávání. Přitom se neberou v úvahu nejvyšší bity jednotlivých znaků, které systém využívá pro uložení tzv. *atributů souboru* (viz např. služba 30 modulu BDOS).

ex – číslo rozšíření

Jedna položka adresáře může popsat soubor omezené velikosti. Proto je soubor rozdělen na tzv. *rozšíření* (extent) o velikosti 16KB. Pokud je celkový počet alokačních bloků na médiu (DSM) menší nebo roven 256, je možno jednotlivé alokační bloky adresovat jednou slabikou. Pro větší média,

kde celkový počet alokačních bloků přesáhne 256, je pro adresu alokačního bloku nutno použít dvě slabiky. V jedné položce adresáře je rezervováno 16 slabik pro adresy alokačních bloků a může zde proto být uloženo 16 nebo 8 adres alokačních bloků. Tato skupina alokačních bloků tvoří jedno rozšíření. Podle velikosti a počtu alokačních bloků je obsah souboru uložen v jednom nebo více rozšířeních.



Obr. 5.3. Soubory tvořené více rozšířeními (soubor ALFA.TXT obsahuje 175 vět délky 128 slabik, uložených ve 22 alokačních blocích rozdělených do dvou rozšíření)

Soubory tvořené více rozšířeními, jsou popsány více položkami adresáře, které mají shodnou identifikaci (uživatele, jméno a typ). Položky mohou být v adresáři uloženy v náhodném pořadí. Aby bylo možno určit pořadí položek, obsahuje každá položka adresáře prvek "ex", kde je uvedeno číslo rozšíření popisovaného touto položkou adresáře. Povolený rozsah hodnot je 0 až 31 (1FH). Je-li číslo rozšíření větší, přetékají vyšší bity do "s2" (viz dále).

s1 – rezervováno
s2 – vyšší bity čísla rozšíření

rc – počet vět délky 128 slabik, využitých v posledním alokačním bloku popisovaném touto položkou

Předchozí alokační bloky jsou vyplněny úplně. Povolený rozsah je 0 až 128 (80H), tj. jeden alokační blok může obsahovat až 128 vět délky 128 slabik (tj. celkem 16 KB). Je-li "rc" menší než 128, předpokládá se, že se jedná o poslední rozšíření souboru.

d0 až d15 – čísla alokačních bloků přidělených souboru

Přenos dat z, resp. do souboru probíhá po větách, ale místo na disku se souboru přiděluje po alokačních blocích. V této části položky adresáře jsou uvedena čísla alokačních bloků přidělených souboru.

V závislosti na hodnotě DSM jsou čísla alokačních bloků uložena na jedné (DSM < 256) nebo dvou (DSM > 256) slabikách. Jedna položka adresáře tedy může popisovat umístění nejvýše 16, resp. 8 alokačních bloků.

Vzhledem k tomu, že nultý alokační blok je vždy adresářový, je 0 v čísle alokačního bloku interpretována jako volné místo pro číslo dalšího bloku.

5.3.4 Správa alokačních bloků

Z uvedené struktury logického diskového média je patrné, že každý alokační blok je buď přidělen nějakému souboru, nebo je volný (není uveden v žádné položce adresáře). Přehled o volných alokačních blocích si systém udržuje pomocí tzv. *bitové mapy alokačních bloků*. Tato bitová mapa není uložena na disku, ale systém CP/M si ji vytváří v paměti při aktivaci disku.

Při první operaci s adresářem diskové jednotky je vytvořena bitová mapa. Místo vyhrazené pro bitovou mapu v tabulkách v operační paměti se nejprve vynuluje a poté jsou v bitové mapě nastaveny jedničky v místech odpovídajících alokačním blokům adresáře (ty jsou vždy obsazeny). Dále je čten adresář a pro všechny alokační bloky platných položek jsou v bitové mapě nastaveny jedničky. Po vytvoření bitové mapy je disková jednotka označena jako aktivní.

Disková jednotka zůstává aktivní až do nového startu systému nebo vyvolání služby inicializace diskového systému. U aktivních jednotek odpovídá bitová mapa stále okamžitému stavu, neboť při přidělení bloku souboru

se příslušný bit nastaví a naopak při uvolnění bloků (rušení souboru) se příslušné bity nulují.

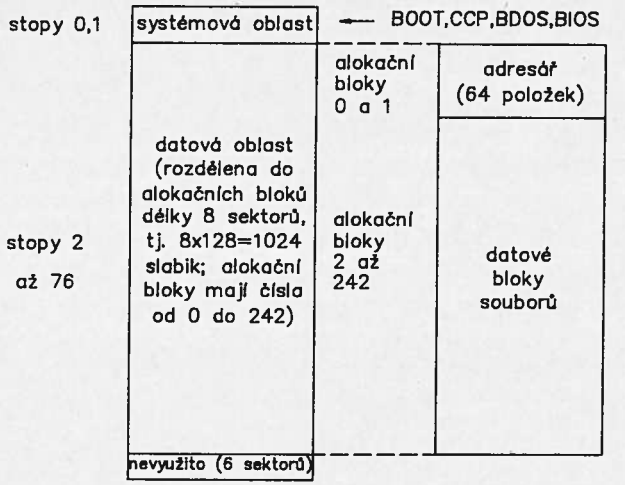
Naproti tomu obsah adresáře nemusí při vytváření souboru odpovídat okamžitému stavu, neboť položka adresáře s čísly přidělených alokačních bloků se zapisuje až po uzavření souboru, resp. při překročení hranice rozšíření. Naopak při rušení souboru je položka na disku označena za neplatnou ihned (zbytek položky zůstává nezměněn). Zrušení souboru se proto projeví na disku ihned, ale zápis do souboru se zafixuje až po uzavření souboru.

Je-li v aktivní jednotce vyměněno médium, neodpovídá bitová mapa skutečnosti. Při zápisu do souboru by proto mohly být přiděleny obsazené alokační bloky a přepsány věty jiného souboru. Aby se takovéto situaci zabránilo, testuje systém u jednotek s výměnnými médii kontrolní součet vět adresáře. Zároveň s vytvářením bitové mapy se počítá součet každé věty adresáře modulo 256 a je zaznamenán do tabulek v paměti. Při dalších přístupech se tento součet kontroluje. V případě nesouhlasu je příslušný disk označen příznakem "jen pro čtení" (R/O). Pokus o zápis je pak hlášen jako chyba, stejně jako při pokusu o zápis na disk, pro který byl zápis explicitně zakázán voláním služby jádra (služba modulu BDOS).

5.3.5 Mapování logického disku na disk fyzický

Věty logického disku mapuje modul BDOS na fyzický disk, tj. na médium členěné do stop a sektorů. Fyzický disk se z hlediska modulu BDOS jeví jako médium s bloky pevné délky 128 slabik, které se identifikují číslem diskové jednotky, číslem stopy a číslem sektoru. Pro práci s fyzickými disky využívá modul BDOS služby SELDSK, SETTRK, SETSEC, READ a WRITE modulu BIOS. Mapování se provádí na základě konstant uvedených pro příslušný logický disk v tabulce DPB (Disk Parameter Block), která je rovněž součástí modulu BIOS.

Logický soubor je chápán jako posloupnost logických vět, bezprostředně po sobě následujících. Logické věty logické diskové jednotky jsou rozděleny do skupin nazývaných logické stopy vždy po SPT větách. Číslo logické stopy je přičtením konstanty OFF převedeno na číslo fyzické stopy. Logickým větám v rámci jedné stopy jsou překladem pomocí tabulky



Obr. 5.4. Rozdělení fyzického disku

(voláním služby SETRAN modulu BIOS) přidělena čísla fyzická. V případě, že délka fyzického sektoru je 128 slabik, odpovídají čísla vět číslům sektorů.

Uvedené transformace nemusí být ještě konečné. Několik logických disků může být simulováno na jednom fyzickém disku vhodnou volbou konstant DSM a OFF. Rovněž logická stopa může být realizována více stopami fyzickými (např. u oboustranné diskety mohou být oba povrchy považovány za jeden s dvojnásobnou kapacitou). Je-li velikost fyzického sektoru větší než 128 slabik, musí se provádět skládání a rozkládání vět do fyzických sektorů pomocí tzv. *blokovacího algoritmu.* Všechny uvedené doplňkové transformace musí být však realizovány modulem BIOS.

5.3.6 Struktura textových souborů

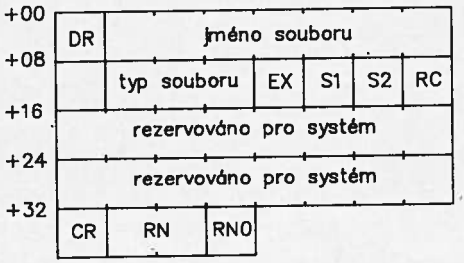
I když vnitřní struktura vět není systémem definována, je vhodné u textových souborů dodržet následující pravidla:

- řádky textového souboru nemusí být stejně dlouhé,
- řádky jsou od sebe odděleny znaky CR (0DH) a LF (0AH) v uvedeném pořadí,
- řádky mohou být zapisovány přes hranice vět,

– pokud poslední věta souboru nekončí přesně na konci věty (tj. znak LF není 128. znakem věty), je zbytek věty vyplněn znaky ^Z (1AH).

5.3.7 Řídicí blok souboru a adresářový kód

Většina služeb operačního systému, které provádějí operace s diskovými soubory, vyžaduje jako parametr adresu zóny v operační paměti o délce 36 slabik. Tato zóna se nazývá *řídicí blok souboru* a označuje se FCB (File Control Block). Hlavní důvod pro zavedení řídicího bloku souboru je urychlení přístupu k obsahu souboru, neboť není nutno odpovídající informace vyhledávat v adresáři na diskovém médiu při každém přístupu k souboru.



Obr. 5.5. Struktura řídicího bloku souboru

Struktura řídicího bloku souboru:

Název	Délka	Význam
dr	1	Číslo diskové jednotky (Drive).
f1 ... f8	8	Jméno souboru (File Name).
t1 ... t3	3	Typ souboru (File Type).
ex	1	Číslo rozšíření (Extent).
s1	1	Rezervováno pro systém.
s2	1	Vyšší bity čísla rozšíření a příznak změny FCB.
rc	1	Počet vět (Record Count).
d0 ... d15	16	Čísla alokačních bloků.
cr	1	Číslo aktuální věty (Current Record).
r0, r1	2	Číslo věty při přímém přístupu (nižší slabika v r0, vyšší v r1).
r2	1	Rezervováno pro systém.

Strukturu řídicího bloku souboru lze rovněž vyjádřit následující deklarací v jazyce Pascal.

```
type
   BYTE      = 0 . . 255;
   FILENAME  = packed array [1 . . 8] of char ;
   FILETYPE  = packed array [1 . . 3] of char ;
   TYPFCB    = packed record  {řídicí blok souboru}
               DR    :BYTE ;     {kód diskové jednotky (0-16)}
               FN    :FILENAME ; {jméno souboru}
               FT    :FILETYPE ; {typ souboru a atributy}
               EX    :BYTE ;     {číslo aktuálního rozšíření (0-31)}
               S1    :BYTE ;     {využívá BDOS}
               S2    :BYTE ;     {využívá BDOS}
               RC    :BYTE ;     {počet vět v rozšíření}
               CPMA:packed array [1 . . 16] of char; {rezervováno}
               CR    :BYTE ;     {číslo věty pro sekvenční přístup}
               RN    :integer ;  {číslo věty pro přímý přístup}
               RNO   :BYTE ;     {vyšší bity RN}
            end;
```

V jazyce symbolických adres používáme následující deklarace pro vyjádření struktury řídicího bloku.

```
; POLOŽKY FCB

FCBDR    EQU  FCB ; OZNAČENÍ DISKOVÉ JEDNOTKY
FCBNAME EQU  FCB+01 ; JMÉNO SOUBORU
FCBTYPE EQU  FCB+09 ; TYP SOUBORU
FCBEX    EQU  FCB+12 ; ČÍSLO AKTUÁLNÍHO ROZŠÍŘENÍ
FCBS2    EQU  FCB+14 ;
FCBRC    EQU  FCB+15 ; POČET VĚT
FCBCR    EQU  FCB+32 ; ČÍSLO AKTUÁLNÍ VĚTY
FCBR0    EQU  FCB+33 ; ČÍSLO SEKTORU PRO PŘÍMÝ
FCBR2    EQU  FCB+35 ; PŘÍSTUP
FCBDR2   EQU  FCB+16 ; JEDNOTKA PRO FCB2
FCBNAM2EQU  FCB+17 ; JMÉNO PRO FCB2
```

Význam položek řídicího bloku:

dr – Číslo diskové jednotky, na které se soubor nalézá:

0 – aktuální disková jednotka
1 – jednotka A
.
.
.
16 – jednotka P

f1 ... f8 – Jméno souboru (doplněno případně mezerami; může obsahovat znaky "?".)

t1 ... 13 – Typ souboru (doplněný případně mezerami; může obsahovat znaky "?". Nejvyšší bity jména a typu souboru mají význam atributů souboru (viz dále služba 30):

nejvyšší bit t1=1 – soubor chráněn proti zápisu (R/O file),
nejvyšší bit t2=1 – systémový soubor (SYS file) není vypisován příkazem DIR,
nejvyšší bit t3=1 – archivní bit (je nastaven při zápisu do souboru).

ex – Číslo aktuálního rozšíření – bity 0 až 4 (tj. povolený rozsah 0 až 1FH)

Vyšší bity jsou v položce s2. Nejvyšší bit s2 je systémem nastaven při aktivaci FCB (otevření rozšíření, vytvoření rozšíření) a nulován při změně FCB (přidělení alokačního bloku, změna rc – počtu zapsaných vět). Při uzavírání rozšíření (explicitním při uzavírání souboru nebo implicitním při přechodu na jiné rozšíření) jsou do adresáře zapisovány pouze změněné řídicí bloky FCB.

rc – Počet vět v posledním alokačním bloku aktuálního rozšíření

d0 ... d15 – Čísla alokačních bloků

Tato položka obsahuje čísla alokačních bloků přidělených aktuálnímu rozšíření souboru. Jestliže celková kapacita diskové jednotky je menší než 257 alokačních bloků, jsou čísla alokačních bloků uložena v jedné slabice,

jinak ve dvou slabikách. Nula na místě čísla alokačního bloku indikuje volnou složku seznamu čísel alokačních bloků, nikoliv alokační blok číslo 0, neboť ten je vždy přidělen adresáři.

cr – Číslo aktuální věty

Číslo věty (v rámci aktuálního rozšíření), která se bude číst nebo zapisovat při sekvenčním přístupu k souboru.

r0, r1 – Číslo věty pro přímý přístup

Při přímém přístupu k větám souboru platí globální číslo logické věty v souboru, uvedené v r0 a r1. Pokud je číslo aktuální věty pro přímý přístup větší než 65535, je využita i položka r2.

r2 – Rezerva

Úvodních 32 slabik řídicího bloku souboru prakticky odpovídá příslušné položce adresáře (slabika dr je v adresáři nahrazena slabikou user, určující číslo uživatele, kterému soubor patří, příp. platnost položky). Je na programu, aby vymezil řídicí bloky souborů, se kterými bude pracovat. Před použitím řídicího bloku musí být nastavena položka "dr" určující diskovou jednotku, jméno a typ souboru (položky "f1" až "f8" a "t1" až "t3"). Položky "ex" a "s2" musí být vynulovány. Pro jeden soubor může být využit implicitní blok FCB vytvořený modulem CCP na adrese 5CH.

Při sekvenčním přístupu k větám souboru je využíváno pouze prvých 33 slabik řídicího bloku, slabiky r0, r1 a r2 nemusí být vůbec přítomny. Číslo aktuální věty (cr) je průběžně aktualizováno při zpracování souboru. Po vyčerpání akutálního rozšíření přejde systém automaticky na další rozšíření a vynuluje číslo aktuální věty. Při přímém přístupu je využito číslo, uvedené v posledních třech slabikách řídicího bloku. Toto číslo se však nezmění při přímém přístupu automaticky, to musí zajistit program.

Systémové služby, které manipulují s adresářem, vracejí jako funkční hodnotu tzv. *adresářový kód* – hodnotu 0, 1, 2, 3 nebo 255. Je-li hodnota adresářového kódu v rozsahu 0 až 3, byla operace úspěšná a hodnota kódu určuje umístění příslušné adresářové položky ve větě adresáře. Hodnota 255 signalizuje chybu – odpovídající položka adresáře nebyla nalezena.

5.4 Služby modulu BDOS

V tomto odstavci se budeme zabývat službami poskytovanými operačním systémem CP/M (přesněji řečeno jeho modulem BDOS) transientním programům, tedy programům zaváděným procesorem příkazů CCP do oblasti transientních programů (oblast TPA, tj. oblast od adresy 100H výše). Může se jednat jak o programy služební (PIP, STAT atd.), tak i uživatelské. Vzhledem k vrstvené architektuře systému CP/M, považujeme za služby systému pouze služby modulu BDOS. Ostatní služby nižších složek architektury (BIOS, technické prostředky) pokládáme za neviditelné.

Logické rozhraní tvořené modulem BDOS proto v podstatě duplikuje některé služby modulu BIOS, aby nebylo nutno se k nim obracet přímo. Týká se to zejména zařízení pro znakový vstup a výstup. Důležitý rozdíl mezi přímým užitím služeb modulu BIOS a služeb modulu BDOS však spočívá v tom, že na úrovni služeb modulu BDOS pracujeme s logickými zařízeními (CON:, RDR:, PUN:, LST:), kterým mohou být přiřazena různá fyzická zařízení. Přesměrování na patřičné služby modulu BIOS zajistí vrstva BDOS. Služby poskytované modulem BDOS lze rozdělit do čtyř skupin:

- služby pro práci s logickými znakovými zařízeními,
- služby pro práci s diskovými jednotkami,
- služby pro práci s diskovými soubory,
- ostatní služby.

Kromě těchto služeb mohou transientní programy využívat informace uložené v komunikační zóně (nulté stránce operační paměti označované jako zóna SPA). Struktura této zóny byla podrobně popsána v odst.5.1.

5.4.1 Konvence volání služeb modulu BDOS

Pro volání služeb modulu BDOS je využita technika jediné pevné adresy – v tomto případě adresy 5. O provedení příslušné služby (funkce) se žádá běžnou instrukcí volání podprogramu (CALL 5), přičemž v registru C se předává číslo požadované služby. Vyžaduje-li služba parametr, uvádí se v registru E, resp. v registrovém páru DE.

V registrech se rovněž předávají informace o výsledcích činnosti služeb. Hodnotu v rozsahu 0 až 255 vrací systém v registru A, větší hodnoty v registrovém páru HL (z důvodu kompatibility se staršími verzemi systému

je vždy současně nastaveno A=L a B=H). Obsah ostatních registrů není definován. Uvedené zásady odpovídají zásadám volání podprogramů v jazyce PL/M.

Při provádění služeb přepíná modul BDOS ukazatel zásobníku na vlastní zásobník, takže nehrozí nebezpečí přeplnění uživatelského zásobníku.

5.4.2 Přehled služeb modulu BDOS

Služby pro obecnou práci se systémem

0	Inicializace systému
12	Dodání čísla verze systému

Služby pro spolupráci se znakovými zařízeními

1	Vstup znaku ze systémové konzole (CON:)
2	Výstup znaku na systémovou konzoli (CON:)
3	Vstup znaku z logického zařízení pro vstup (RDR:)
4	Výstup znaku na logické zařízení pro výstup (PUN:)
5	Výstup znaku na logické zařízení pro tisk (LST:)
6	Přímý vstup nebo výstup na systémovou konzoli
7	Dodání aktuální hodnoty slabiky IOBYTE
8	Nastavení aktuální hodnoty slabiky IOBYTE
9	Výstup řetězu znaků na systémovou konzoli
10	Čtení editovaného řádku ze systémové konzole
11	Test stavu systémové konzole

Služby pro práci s diskovými jednotkami

13	Inicializace diskového systému
14	Nastavení aktuální diskové jednotky
15	Dodání vektoru aktivních diskových jednotek
25	Dodání čísla aktuální diskové jednotky
27	Dodání adresy bitové mapy
28	Zákaz zápisu na aktuální diskovou jednotku
29	Dodání vektoru diskových jednotek se zákazem zápisu
31	Dodání adresy tabulky popisu disku DPB
37	Inicializace vybrané diskové jednotky

Služby pro práci se soubory

15	Otevření souboru
16	Uzavření souboru
17	Vyhledání prvního výskytu souboru
18	Vyhledání dalšího výskytu souboru
19	Zrušení souboru
20	Sekvenční čtení věty
21	Sekvenční zápis věty
22	Vytvoření souboru
23	Přejmenování souboru
26	Nastavení adresy DMA
30	Nastavení atributů souboru
32	Nastavení, příp. dodání čísla aktuálního uživatele
33	Čtení věty s přímým přístupem
34	Zápis věty s přímým přístupem
35	Výpočet velikosti souboru
36	Nastavení čísla věty pro přímý přísup
40	Zápis věty s přímým přístupem s doplněním nul

Poznámka: služby 7, 8, 28 a 32 nejsou dostupné ve variantě MP/M. Ve verzi 3.1 je služba 7 nahrazena službou čtení stavu zařízení pro logický vstup, služba 8 je nahrazena službou čtení stavu zařízení pro logický výstup. Slabika IOBYTE není ve verzích 3.1 a vyšších implementována a přes služby modulu BDOS jsou dostupná pouze logická zařízení.

5.4.3 Služby pro obecnou práci se systémem

Modul BDOS poskytuje programům dvě služby na obecné úrovni komunikace se systémem.

Služba 0: Inicializace systému

vstup: C – 00H
výstup: není (bez návratu)

Služba 0 provádí *inicializaci modulu BDOS* a je použitelná pro ukončení programu a předání řízení systému, pokud nebyl rezidentní kód systému

činností programu porušen. Vyvolání služby v podstatě odpovídá teplému startu (skok na adresu 0). Po inicializaci modulu BDOS je ze systémové jednotky do paměti zaveden modul CCP. Služba dále čte adresáře aktivních jednotek (aktivuje je), označí aktivované diskové jednotky jako přístupné pro čtení i zápis, nastaví implicitní adresu DMA (tj. 80H) a aktivuje procesor příkazů CCP. Jako aktuální je nastavena disková jednotka A. Oproti teplému startu systému není přenášen do paměti modul BDOS, proto je použití této služby možné pouze tehdy, není-li modul BDOS programem porušen.

Služba 12: Dodání čísla verze systému

vstup: C – 0CH
výstup: HL – číslo verze systému

Služby systému se mírně liší v různých verzích. Ke zjištění aktuální verze systému proto poskytuje modul BDOS službu 12, která vrátí v registrovém páru HL *číslo verze systému*. Na základě znalosti konkrétní verze systému pak může program modifikovat svou činnost. Číslo verze je kódováno tak, že v registru H vrací služba hodnotu 0 pro monoprogramové verze systému CP/M a hodnotu 1 pro systémy MP/M. V registru L se vrací hodnota 0 pro verze nižší než 2.0, hodnota 20H pro verzi 2.0, 31H pro verzi 3.1 atd.

5.4.4 Služby pro spolupráci se znakovými zařízeními

Služby modulu BDOS pro spolupráci se znakovými zařízeními jsou na různé úrovni. Poměrně bohatý je repertoár služeb pro spolupráci se systémovou konzolí. Pro ostatní logická zařízení existují jen velmi primitivní služby. Verze 2.2 a nižší dále umožňují programům číst a modifikovat aktuální přiřazení logických a fyzických zařízení pomocí obsahu slabiky IOBYTE.

Služby pro spolupráci se systémovou konzolí

Přestože je spolupráce se systémovou konzolí z logických zařízení nejlépe obsloužena, zůstává na úrovni fyzické komunikace. Zejména při výstupu na systémovou konzoli chybí služby vyšší úrovně, např. služba pro vymazání obsahu obrazovky, nezávislá na konkrétní realizaci systémové konzole.

Systémovou konzoli lze pomocí služeb modulu BDOS ovládat na třech úrovních. Nejnižší je *služba pro přímý vstup a výstup* na systémovou konzoli. Představuje v podstatě přímé volání služeb modulu BIOS. Je určena pro programy nekomunikující se systémovou konzolí řádkovým způsobem (např. obrazovkový editor nebo programy ovládané jednotlivými klávesami).

Služba 6: Přímý vstup nebo výstup na systémovou konzoli

vstup: C – 06H
E – 255 nebo znak
výstup: A – 0 nebo znak (bylo-li E=255)

Tato služba zajišťuje přímou komunikaci (*vstup* nebo *výstup*) se systémovou konzolí bez kontroly přenášených znaků systémem (nekontrolují se znaky ^S, ^P ani ^C). Je-li v registru E zadána hodnota 255 (0FFH), je dodán jeden znak přečtený ze systémové konzole v registru A, příp. je v registru A vrácena hodnota 0, není-li znak k dispozici. Pokud nebyla v registru E zadána hodnota 255, je obsah registru E vyslán na systémovou konzoli. Služba 6 nečeká na stisk znaku na klávesnici konzole, ale sama vrací řízení volajícímu programu. Nedoporučuje se ji kombinovat s jinými službami pro ovládání systémové konzole, neboť by mohlo dojít k nežádoucím interferencím.

Druhou úroveň služeb pro práci se systémovou konzolí představuje sada služeb pro komunikaci na úrovni jednotlivých znaků:
– zjištění stavu systémové konzole,
– vstup znaku ze systémové konzole a
– výstup znaku na systémovou konzoli.

Služba 11: Test stavu systémové konzole

vstup: C – 0BH
výstup: A – 255 (pokud byla stisknuta klávesa), jinak 0

Test stavu systémové konzole zajišťuje služba 11. Tato služba testuje stisk klávesy na systémové konzoli. Byla-li stisknuta některá klávesa, vrací se v registru A hodnota 255 (0FFH), jinak vrací služba hodnotu 0.

Služba 1: Vstup znaku ze systémové konzole

vstup: C – 01H
výstup: A – přečtený znak

Pro *vstup znaku* ze systémové konzole je určena služba 1. Znak přečtený ze systémové konzole je předán v registru A. Služba čeká na stisk znaku, pokud již není připraven. Přečtený znak je zobrazen na systémové konzoli, s výjimkou řídicích znaků. Z řídicích znaků jsou zaslány na systémovou konzoli pouze znaky CR (0DH), LF (0AH), BS (backspace = ^H) a tabelátor (^I). Znak tabelátor je při tom nahrazen patřičným počtem mezer. Ve vstupujících znacích se testují znaky ^S (výstup se pozastaví do stisku dalšího znaku) a ^P (střídavě zapíná a vypíná kontrolní opis výstupu na systémové konzoli na logickém zařízení pro tisk).

Služba 2: Výstup znaku na systémovou konzoli

vstup: C – 02H
E – znak
výstup: není

Výstup znaku na systémovou konzoli zprostředkuje služba 2. Znak předaný v registru E je zobrazen na systémové konzoli. Podobně jako u služby 1 jsou tabelátory nahrazeny mezerami. Při výstupu znaku se testuje, nebyla-li na systémové konzoli stisknuta klávesa. Jestliže ano, je znak přečten. Jedná-li se o znak ^S, je výstup pozastaven až do stisku další, libovolné klávesy. Jak znak ^S, tak i znak uvolňující výstup jsou zapomenuty. Pokud byla stisknuta klávesa ^P, zapne se nebo vypne kontrolní opis. Byla-li stisknuta jiná klávesa než ^S nebo ^P, je znak uschován a předán při následující žádosti o vstup ze systémové konzole (volání služby 1 nebo 10). Až do předání tohoto znaku nebo nového startu systému se testování vstupu neprovádí.

Pro výstup speciálních řídicích znaků na systémovou konzoli je někdy třeba nastavit nejvyšší bit, aby např. binární kombinace 09H nebyla považována za tabelátor. Rovněž je možno použít službu 6.

Nejvyšší úroveň spolupráce se systémovou konzolí poskytují služby pro práci s řetězy znaků.

Služba 9: Výstup řetězu znaků na systémovou konzoli

vstup: C – 09H
DE – adresa řetězu znaků ukončeného znakem "$"
výstup: není

Výstup řetězu znaků na systémovou konzoli zajišťuje služba 9. Jako parametr se zadává adresa řetězu znaků v registrovém páru DE. Řetěz znaků zakončený znakem "$" je vypsán na systémové konzoli (znak "$" se nevypisuje). Pro výpis řetězu platí stejná pravidla jako pro výstup jednotlivých znaků pomocí služby 2, tj. kontrolují se speciální klávesy.

Služba 10: Čtení editované řádky ze systémové konzole

vstup: C – 0AH
DE – adresa zóny v paměti
výstup: znaky uložené v zóně paměti

Služba zajistí *přečtení řádku znaků* ze systémové konzole do vyrovnávací paměti, jejíž adresa se zadává v registrovém páru DE. První slabika vyrovnávací paměti musí obsahovat maximální povolený počet znaků (délka vyrovnávací paměti – minimálně 2 slabiky, maximálně 255 slabik). Po provedení služby 10 obsahuje druhá slabika skutečný počet přečtených znaků, které jsou uloženy ve vyrovnávací paměti počínaje od třetí slabiky. Vstup je ukončen znakem CR (který ukládán není) nebo naplněním vyrovnávací paměti. Při vstupu jsou interpretovány řídicí znaky (odst. 4.1) i speciální znaky ^C (teplý start systému – indikován jen v první pozici řádku) a ^P (zapnutí nebo vypnutí opisu znaků vystupujících na konzoli na zařízení pro tisk). V zadávaném řádku je možno editovat pomocí editačních kláves (odst.4.1).

Služby pro spolupráci s logickými zařízeními

Ostatní logická znaková zařízení ovládají služby, které jsou fakticky transparentní směrem k modulu BIOS.

Služba 3: Vstup znaku ze zařízení znakového vstupu

vstup: C – 03H
výstup: A – přečtený znak

Služba 3 dovoluje *vstup znaku* ze zařízení znakového vstupu. Jeden znak z logického zařízení znakového vstupu (logického zařízení RDR:) je předán v registru A. Pokud není logické zařízení pro vstup přítomno, je obvykle přečten znak konec souboru (znak EM nebo ^Z, tj. 1AH). Služba

vždy čeká, dokud není přečten znak, což může způsobit obtíže, pokud není znak např. na komunikační lince připraven. Přitom nelze přímo zjistit stav logického zařízení pro vstup (neexistuje obdoba služby 11). Někdy se tento problém obchází připojením komunikační linky jako zařízení TTY: k systémové konzoli, kde stav vstupu lze testovat.

Služba 4: Výstup znaku na zařízení znakového výstupu

vstup: C – 04H
E – znak
výstup: není

Výstup znaku na zařízení znakového výstupu zprostředkuje služba 4, která zajistí vyslání znaku z registru E na logické zařízení znakového výstupu (PUN:). Přestože mohou nastat podobné problémy jako u služby 3, nedojde zde k zastavení činnosti, neboť služba bezprostředně po zaslání znaku na výstup vrací řízení volajícímu programu.

Služba 5: Výstup znaku na logické zařízení pro tisk

vstup: C – 05H
E – znak
výstup: není

Pro výstup znaku na logické zařízení pro tisk je určena služba 5, která znak uložený v registru E vyšle na logické zařízení pro tisk (LST:). Opět mohou nastat problémy, pokud zařízení pro tisk nepracuje správně. Tyto problémy však musí být řešeny na úrovni modulu BIOS.

Služby pro přiřazení logických a fyzických zařízení

Služba 7: Dodání aktuálního stavu IOBYTE

vstup: C – 07H
výstup: A – aktuální hodnota IOBYTE

Služba 7 vrací v registru A aktuální stav slabiky IOBYTE, který určuje aktuální přiřazení logických a fyzických zařízení (pokud je tato služba implementována modulem BIOS, příp. procesorem příkazů systému).

Služba 8: Nastavení aktuální hodnoty IOBYTE

vstup: C – 08H
E – nová hodnota IOBYTE
výstup: není

Služba 8 nastaví *aktuální přiřazení* fyzických zařízení logickým, tj. obsah slabiky IOBYTE (obsah adresy 3) na hodnotu uvedenou v registru E.

5.4.5 Služby pro práci s diskovými jednotkami

Modul BDOS umožňuje spolupráci s vnějšími pamětmi na dvou úrovních. Prvá úroveň se týká ovládání diskových jednotek, druhá pak realizuje systém ovládání souborů na vnějších pamětech.

Na úrovni ovládání diskových jednotek jsou k dispozici následující třídy služeb:
- služby pro inicializaci diskových jednotek,
- služby pro ovládání aktuální jednotky,
- služby pro aktivaci diskových jednotek,
- služby pro řízení zákazu zápisu na jednotky a
- služby pro zjištění parametrů jednotky.

Služby pro inicializaci diskových jednotek

Služba 13: Inicializace diskového systému

vstup: C – 0DH
výstup: není

Inicializaci celého *diskového systému* zajišťuje služba 13, kterou často využívají transientní programy při zahájení činnosti. Tato služba uvádí diskový systém do počátečního stavu. Aktivní je pouze systémová disková jednotka A, jenž je zároveň jednotkou aktuální. U všech diskových jednotek je zrušen zákaz zápisu a adresa DMA je nastavena na implicitní hodnotu (80H).

Služba 13 je využívána i modulem CCP při startu systému a využívají ji též programy, které vyžadují výměnu diskových médií bez inicializace celého operačního systému. Kromě inicializace celého diskového systému lze inicializovat pouze vybrané diskové jednotky pomocí služby 37.

Služba 37: Inicializace vybraných diskových jednotek

vstup: C – 25H
DE – bitová mapa inicializovaných jednotek
výstup: A = 0

Na rozdíl od služby 13 provede služba 37 pouze inicializaci diskových jednotek zadaných ve formě bitové mapy v registrovém páru DE. Nejnižší bit registru E (bit 0) popisuje jednotku A, nejvyšší bit registru D (bit 7) popisuje jednotku P. Jednička v příslušném bitu znamená, že odpovídající jednotka má být inicializována. Adresa zóny DMA ani aktuální jednotka se při použití služby 37 nemění. Služba vrací v registru A hodnotu 0 kvůli kompatibilitě se systémem MP/M.

Služby pro ovládání aktuální jednotky

Služba 14: Nastavení aktuální diskové jednotky

vstup: C – 0EH
E – číslo jednotky
výstup: není

Nastavení *aktuální diskové jednotky* umožňuje služba 14. Jednotka, jejíž číslo je zadáno v registru E (0=A, 1=B, ... , 15=P), je aktivována (pokud dosud nebyla) a stane se jednotkou aktuální. Její číslo je uloženo do komunikační zóny (na nižší bity adresy 4). K aktuální jednotce se vztahují diskové operace, pro které je v příslušném řídicím bloku souboru uvedena ve slabice dr (drive) hodnota 0.

Služba 25: Dodání čísla aktuální diskové jednotky

vstup: C – 19H
výstup: A – číslo aktuální diskové jednotky

Nastavenou *aktuální diskovou jednotku* lze programově zjistit pomocí služby 25. Číslo aktuální diskové jednotky (0=A, 15=P) je dodáno v registru A. Pro aktuální diskovou jednotku existuje možnost získat stav zaplnění média.

Služba 27: Dodání adresy bitové mapy aktuální jednotky

vstup: C – 1BH
výstup: HL – adresa bitové mapy

Služba 27 vrací v registrovém páru HL adresu *začátku bitové mapy* (alokačního vektoru) aktuální diskové jednotky. Bitová mapa je analyzována např. programem STAT při výpočtu volného místa na disku. Rovněž modul BDOS využívá této služby při zjišťování volného místa na disku, nebo při alokování dalšího bloku pro soubor. Uživatelskými programy nebývá tato služba využívána.

Služby pro aktivaci diskových jednotek

Služba 24: Dodání vektoru aktivních diskových jednotek

vstup: C – 18H
výstup: HL – vektor aktivních jednotek

Programovými službami lze rovněž ovládat aktivaci diskových jednotek. Služba 24 dodá v registrovém páru HL *vektor aktivních diskových jednotek.* Jednotce A odpovídá nejnižší bit registru L (bit 0), jednotce P nejvyšší bit registru H (bit 7). Aktivní jednotky jsou indikovány jedničkovými bity. Aktivovat zvolenou jednotku je možno pomocí dočasného přepnutí aktuální diskové jednotky přes služby pro nastavení aktuální jednotky.

Služby pro řízení zákazu zápisu na jednotky

Služba 28: Zákaz zápisu na aktuální disk

vstup: C – 1CH
výstup: není

Pokus o *zápis na diskovou jednotku*, na kterou byl službou 28 zápis zakázán, způsobí vypsání chybové zprávy na systémové konzoli

BDOS Error on d:R/O

kde d je označení příslušné diskové jednotky. Zákaz zápisu nastavený voláním služby 28 platí, stejně jako zákaz zápisu nastavený systémem při

zjištění nedovolené výměny média, až do příštího studeného nebo teplého startu nebo inicializace diskového systému (služba 13).

Služba 29: Dodání vektoru diskových jednotek se zákazem zápisu

vstup: C – 1DH
výstup: HL - vektor diskových jednotek se zákazem zápisu

Služba 29 vrací v registrovém páru HL bitový *vektor diskových jednotek* se zákazem zápisu (R/O). Jedničky v bitech registrového páru HL určují diskové jednotky, na kterých je zakázán zápis tak, že nejnižší bit (bit 0) registru L odpovídá jednotce A, nejvyšší bit registru H (bit 7) jednotce P. Zápis byl zakázán buď programově voláním služby 28, nebo zjištěním nedovolené výměny média v aktivní jednotce.

Služby pro zjištění parametrů jednotky

Služba 31: Dodání adresy tabulky DPB

vstup: C – 1FH
výstup: HL – adresa tabulky DPB

Aby bylo možno programově zjistit parametry diskových jednotek uložené v modulu BIOS, poskytuje modul BDOS programům službu 31, která vrací v registrovém páru HL *adresu tabulky DPB* (Disk Parametr Block) obsahující parametry aktuální diskové jednotky (viz popis tabulek modulu BIOS). Tato služba je využívána např. programem STAT při výpisu charakteristik diskových jednotek. Dále může být použita při dynamické změně parametrů diskové jednotky (např. při přepínání z jednoduché hustoty záznamu na dvojitou).

5.4.6 Služby pro práci se soubory

Realizace systému ovládání souborů představuje logicky nejvyšší úroveň služeb modulu BDOS. Služby pro práci se soubory zakrývají programům podrobnosti spolupráce s fyzickou strukturou diskových médií a fyzickým ovládáním diskových jednotek. Definice jednotné organizace uložení dat na diskových médiích dovoluje sdílení diskových médií mezi různými programy.

Každý program by principiálně mohl řídit spolupráci s diskem sám, ale výhodnější je zabudovat tyto funkce jako služby přímo do jádra systému.

Služby modulu BDOS vytvářejí zdání existence logických diskových jednotek, na nichž lze pracovat se soubory. Informace potřebné pro transformaci logické struktury na fyzickou jsou uloženy v adresáři diskového média. Každý přístup k obsahu souboru by proto vyžadoval přečtení a příp. aktualizaci adresáře média. Počet přístupů na diskové médium lze značně zredukovat, pokud během spolupráce se souborem udržujeme odpovídající informace v operační paměti.

Pro tento účel zavádí operační systém CP/M zónu v operační paměti nazývanou *řídicí blok souboru* (FCB – File Control Blok). Při zahájení práce se souborem zkopírujeme do této zóny informace z adresáře diskového média. Tato akce se nazývá *otevření souboru*. Před voláním služeb modulu BDOS musíme v příslušném řídicím bloku souboru nastavit vhodné informace. Následující podprogram ilustruje inicializaci řídicího bloku.

```
; PODPROGRAM INIFCB
;--------------
; PODPROGRAM INIFCB SLOUŽÍ PRO INICIALIZACI FCB.
; PARAMETRY:
; VSTUP: HL - ADRESA ŘÍDICÍHO BLOKU FCB.
;        DE - ADRESA ZÓNY, KTERÁ OBSAHUJE
;             IDENTIFIKACI SOUBORU VE
;             STANDARDNÍM TVARU:
;                D : NNNNNNNN . TTT, KDE:
;                D ..... DISKOVÁ JEDNOTKA,
;                NNNNNNNN . . JMÉNO SOUBORU A
;                TTT ..... TYP SOUBORU
;             DÉLKA JE PŘESNĚ 14 SLABIK, JMÉNO A TYP
;             SOUBORU MUSÍ BÝT DOPLNĚNO NA
;             PŘÍSLUŠNOU DÉLKU MEZERAMI.
; VÝSTUP: INICIALIZOVANÝ BLOK FCB NA ADRESE V HL.
; VOLÁNÍ:
;       LD    DE,NAME; ADRESA JMÉNA SOUBORU
;       LD    HL,FCB   ; ADRESA ŘÍDICÍHO BLOKU
;       CALL  INIFCB
```

```
INIFCB: PUSH  BC      ; USCHOVEJ REGISTRY BC.
LD      A,(DE)        ; PŘESUN JMÉNA SOUBORU DO
                      ; FCB.
SUB     'B'           ; KÓDOVÁNÍ JEDNOTEK JE:
                      ; 0=AKTUÁLNÍ, 1=A, 2=B, ATD.
LD      (HL),A        ; NASTAV JEDNOTKU V FCB
INC     DE            ; PŘESKOK JEDNOTKY
INC     DE            ; PŘESKOK ":"
EX      DE,HL
INC     DE
LD      BC,8
LDIR                  ; PŘESUN JMÉNA DO FCB
INC     HL            ; PŘESKOK "."
LD      BC,3
LDIR                  ; PŘESUN TYPU DO FCB
LD      A,0           ; DALŠÍ INICIALIZACE
LD      (DE),A        ; NASTAV EX
INC     DE
INC     DE
LD      (DE),A        ; NASTAV S2
POP     BC            ; OBNOV REGISTRY BC
RET
```

Služby pro manipulaci se soubory

Služba 15: Otevření souboru

vstup: C – 0FH
DE – adresa FCB
výstup: A – adresářový kód

Modul BDOS poskytuje pro *otevření souboru* službu 15, které jako parametr předáme v registrovém páru DE adresu řídicího bloku. Službou 15 se aktivuje řídicí blok souboru, pokud soubor v adresáři příslušné jednotky existuje. Před voláním je třeba v řídicím bloku vyplnit označení diskové jednotky (položka dr), jméno a typ souboru (f1 až f8, t1 až t3) a vynulovat položky ex a s2 (položku s1 nuluje systém). Bude-li se soubor zpracovávat

sekvenčním přístupem od prvé věty, je rovněž třeba vynulovat položku cr. Jestliže soubor uvedeného jména neexistuje, ohlásí služba 15 chybu tím, že v registru A vrátí hodnotu 255 (0FFH).

Aktivace řídicího bloku spočívá ve vyhledání položky v adresáři oblasti aktuálního uživatele zadané jednotky, která se shoduje s řídicím blokem ve slabikách 1 až 14 a okopírování položky adresáře do řídicího bloku.

Jméno a typ souboru v řídicím bloku může obsahovat i malá písmena. Takto lze programově přistupovat i k souborům, které jsou prostřednictvím procesoru příkazů CCP nepřístupné (neboť CCP překládá malá písmena příkazové řádky na velká).

Jsou-li jméno nebo typ souboru nejednoznačné (obsahují znaky "?"), otevře se první nalezený soubor, jehož položka v adresáři se liší pouze v místech znaku "?". Tento postup nelze doporučit, neboť jeho efekt závisí na momentálním stavu adresáře. K vyhledávání jmen souborů odpovídajících nejednoznačnému jménu slouží služby 17 a 18, které naleznou odpovídající položku v adresáři, a otevření souboru lze provést po náhradě nejednoznačného jména v řídicím bloku jménem jednoznačným.

Při porovnávání jména a typu souboru se srovnává pouze spodních 7 bitů jednotlivých slabik jména a typu. Nejvyšší bity mají význam atributů souboru (viz služba 30).

Pomocí více řídicích bloků lze jeden soubor otevřít vícenásobně. Několikanásobně otevřený soubor lze pouze číst. Kombinace čtení a zápisu nebo několikanásobný zápis neproběhne správně, i když chyba se neohlásí. Následující program ilustruje využití služby otevření souboru.

```
BDOS  EQU   5
      ; INICIALIZACE ŘÍDICÍHO BLOKU FCB
      LD    HL, FCB
      LD    DE, NAME
      CALL  INIFCB
      LD    C, 15       ; OTEVŘI SOUBOR
      LD    DE, FCB
      CALL  BDOS
      CP    255
      JP    Z, CHYBA; NELZE OTEVŘÍT
      ; SOUBOR OTEVŘEN
```

Služba 16: Uzavření souboru

vstup: C – 10H
DE – adresa FCB
výstup: A – adresářový kód

Informace, udržované v řídicím bloku souboru v operační paměti, nejsou průběžně zapisovány na disk při změnách v souboru. Tím by se znehodnotil přínos zavedení řídicího bloku. Při ukončení spolupráce se souborem je však nutno provést aktualizaci informace v adresáři diskového média. Tato akce je nazývána *uzavření souboru*. Modul BDOS pro ni poskytuje službu 16, které samozřejmě předáme jako parametr v registrovém páru DE adresu řídicího bloku uzavíraného souboru. Výstupní parametry jsou stejné jako u služby 15.

Uzavírat je nutno pouze soubory, do kterých se zapisovalo. Vzhledem k tomu, že uzavření souboru, ze kterého se pouze četlo, nevyvolává fyzické diskové operace, a vzhledem ke kompatibilitě s operačním systémem MP/M, se doporučuje uzavírat všechny soubory.

Uzavření souboru, do kterého se zapisovalo, vyvolá aktualizaci příslušné položky adresáře podle obsahu řídicího bloku. V žádném případě však nedojde k zápisu věty. I poslední (třeba jen částečně zaplněná věta) musí být před uzavřením souboru zapsána programem.

Není-li výstupní soubor uzavřen, není poslední rozšíření souboru zaneseno v adresáři a tudíž data souboru jím specifikovaná jsou ztracena. Po inicializaci příslušné diskové jednotky (startu systému) jsou alokační bloky přidělené tomuto rozšíření považovány za volné. Použití služby 16 ilustruje následující program.

```
BDOS  EQU  5

      LD     C, 16
      LD     DE, FCB
      CALL   BDOS
      CP     255
      JP     Z, CHYBA; NELZE UZAVŘÍT
      ; SOUBOR UZAVŘEN
```

Před vlastním otevřením souboru pro zpracování je nutno ověřit, zda vůbec soubor s odpovídající identifikací existuje. Pro tento účel poskytuje modul BDOS dvě služby – vyhledání prvého výskytu souboru (služba 17) a vyhledání dalšího výskytu souboru (služba 18). Nutnost zavedení dvojice služeb je vyvolána možností využít nejednoznačné identifikace souboru – označení skupiny souborů. Procházení všech členů skupiny nelze zajistit jednou službou, neboť by se vždy vyhledal prvý člen skupiny.

Služba 17: Vyhledání souboru

vstup: C – 11H
DE – adresa FCB
výstup: A – adresářový kód

Služba 17 zajišťuje *vyhledání prvého výskytu souboru*, jehož identifikace (příp. nejednoznačná, tj. obsahující znaky "?") je zadána v řídicím bloku. Adresa řídicího bloku s příslušnou identifikací se předává jako parametr v registrovém páru DE. Služba prohledává adresář disku (určeného slabikou dr v řídicím bloku) v oblasti aktuálního uživatele a vyhledává položku shodující se s řídicím blokem v pozicích 1 až 14 (pozice obsahující znak "?" se neporovnávají). Je-li nalezena shodná položka, je příslušná věta adresáře přečtena do aktuální oblasti DMA (viz služba 26). Adresářový kód vrácený službou 17 v registru A určuje umístění položky adresáře v rámci věty. Není-li shodná položka nalezena, ohlásí služba chybu tím, že do registru A uloží hodnotu 255 (0FFH).

Je-li znak "?" uveden ve slabice dr řídicího bloku, je prohledáván aktuální disk a dodána každá (i neplatná) položka každého uživatele. Použití služby 17 ilustruje následující program.

```
; INICIALIZACE ŘÍDICÍHO BLOKU
LD      HL, FCB   ; PŘESUN IDENTIFIKACE SOUBORU
CALL    MOVNAM
LD      A,0       ; DALŠÍ INICIALIZACE
LD      (FCB+FCBEX),A  ; NASTAV EX
LD      (FCB+FCBS2),A  ; NASTAV S2
LD      C,17
LD      DE, FCB   ; HLEDEJ, ZDA SOUBOR EXISTUJE
```

```
CALL  BDOS
CP    255
JP    Z,CHYBA ; POKUD NEEXISTUJE —> CHYBA
; SOUBOR EXISTUJE
```

Služba 18: Vyhledání dalšího výskytu souboru

vstup: C – 12H
výstup: A – adresářový kód

Služba 18 doplňuje službu 17 a *vyhledává další výskyt souboru* srovnatelný s identifikací v řídicím bloku. Volání služby 18 musí bezprostředně následovat po volání služby 17 (mezi voláním služeb 17 a 18 nesmí být vyvolána žádná operace s adresářem). Prohledávání adresáře pokračuje od naposledy nalezené položky (službou 17 nebo 18).

Služba 18 nemá vstupní parametr, ale pracuje s řídicím blokem souboru zadaným při předchozím volání služby 17. Tento řídicí blok se proto nesmí modifikovat, má-li být činnost služby 18 správná.

Je-li pro vyhledané soubory třeba provádět akci manipulující s adresářem (kopírování, rušení apod.), lze zvolit jeden z následujících postupů:

a) Vyhledat všechny soubory, uložit jejich jména do tabulky a teprve potom provádět požadované akce.

b) První soubor vyhledat pomocí služby 17. Jeho jednoznačné jméno přenést ze zóny DMA do řídicího bloku FCB a provést požadovanou akci. Další soubory se zpracují tak, že se pro řídicí blok FCB s posledním jednoznačným jménem provede služba 17, v FCB se obnoví nejednoznačné jméno souboru a vyvolá se služba 18. Je-li nalezen další soubor, přenese se jeho jméno ze zóny DMA do FCB a činnost se opakuje.

Služba 22: Vytvoření souboru

vstup: C – 16H
DE - adresa FCB
výstup: A – adresářový kód

K zajištění dynamického života souborů je nutno mít k dispozici prostředky pro vytváření a rušení souborů. Služba 22 modulu BDOS zajišťuje *vytvoření souboru*, jehož identifikaci uložíme do řídicího bloku. Tato služba

je podobná službě 15 (otevření souboru) s tím rozdílem, že v řídicím bloku FCB musí být uvedeno jednoznačné jméno souboru, který doposud v adresáři neexistuje.

Systém vytvoří v adresáři položku odpovídající prázdnému souboru a zároveň inicializuje řídicí blok FCB jako při otevření souboru. Pokud není možno novou položku v adresáři vytvořit (adresář je zaplněn), signalizuje služba 22 chybu hodnotou 255 (0FFH) v registru A. V opačném případě vrací služba v registru A polohu položky ve větě adresáře – adresářový kód.

Systém nekontroluje, nevytváří-li se soubor, který již v adresáři existuje. Hrozí-li možnost takovéto kolize, je třeba pro stejný řídicí blok FCB nejprve vyvolat službu 19 (zrušení souboru) a ignorovat jí vrácený adresářový kód (tj. pokračovat, ať již se soubor našel a zrušil, či nikoliv). V následujícím příkladu je uveden postup při vytváření nového souboru pomocí služeb modulu BDOS.

```
        ; INICIALIZACE ŘÍDICÍHO BLOKU
        LD      HL, FCB
        LD      DE, NAME
        CALL    INIFCB
        LD      C, 17
        LD      DE, FCB         ; HLEDEJ, ZDA SOUBOR
                                ; EXISTUJE
        CALL    BDOS
        CP      255
        JP      Z, CREATE       ; POKUD NEEXISTUJE —>
                                ; LZE VYTVOŘIT
        JP      CHYBA           ; SOUBOR EXISTUJE
CREATE: LD      DE, FCB         ; VYTVOŘENÍ NOVÉHO
                                ; SOUBORU
        LD      C, 22
        CALL    BDOS
        CP      255
        JP      Z, CHYBA        ; SOUBOR EXISTUJE
        ; SOUBOR VYTVOŘEN
```

Služba 19: Zrušení souboru

vstup: C – 13H
DE – adresa FCB
výstup: A – adresářový kód

Další službou pro manipulaci se soubory je služba 19, která umožňuje *zrušení souboru*. Služba 19 zruší soubor (soubory) určené jménem uvedeným v řídicím bloku souboru FCB. Jméno souboru může být nejednoznačné, číslo jednotky (dr) nikoliv. Chyba nastane, nebyl-li zrušen žádný soubor a služba pak vrátí v registru A hodnotu 255 (0FFH). Pro každý nalezený soubor jsou zrušena všechna jeho rozšíření bez ohledu na obsah slabiky ex (není možno rušit jen některá rozšíření souboru).

```
; INICIALIZACE ŘÍDICÍHO BLOKU
LD      HL, FCB
LD      DE, NAME
CALL    INIFCB
LD      C, 17
LD      DE, FCB          ; HLEDEJ, ZDA SOUBOR EXISTUJE
CALL    BDOS
CP      255
JP      Z, CHYBA         ; POKUD NEEXISTUJE —> CHYBA
LD      C, 19            ; JINAK ZRUŠ SOUBOR
LD      DE, FCB2
CALL    BDOS
```

Služba 23: Přejmenování souboru

vstup: C – 17H
DE – adresa FCB
výstup: A – adresářový kód

Změnu *identifikace souboru* v rámci jednoho diskového média zajišťuje služba 23, která přejmenuje soubor, jehož původní jméno a typ je uvedeno v řídicím bloku FCB. Nové jméno a typ souboru je uvedeno ve stejném řídicím bloku, ale v pozicích 17 až 27 (8 znaků jméno, 3 znaky typ, příp. doplněné mezerami). Není-li soubor v adresáři disku určeného

položkou dr nalezen, je do registru A uložena hodnota 255 (0FFH). V opačném případě obsahuje registr A adresářový kód.

Systém nekontroluje, jestli vznikne přejmenováním soubor, který již v adresáři existuje. Proto je před přejmenováním vhodné nejprve vyvolat službu 17 (vyhledání souboru) nebo 15 (otevření souboru), které však předložíme v registrovém páru DE adresu FCB+16. Teprve když služba 17 vrátí hodnotu 255, je možno vyvolat službu 23. Po vyvolání služby 23 je slabika FCB+16 vynulována.

Přejmenovávat nelze soubory s nastaveným příznakem "jen pro čtení" (R/O). Je-li přejmenování úspěšné, jsou zároveň nastaveny atributy souboru podle příznaků uvedených v novém jménu souboru.

```
LD      HL, FCB          ; PŘESUŇ STARÉ - JMÉNO DO FCB
LD      DE, OLDNAME
CALL    INIFCB
LD      HL, FCBDR2       ; NOVÉ JMENO DO FCB2
LD      DE, NEWNAME
CALL    INIFCB
LD      C, 15
LD      DE, FCBDR2       ; POKUSNĚ OTEVŘI SOUBOR
CALL    BDOS             ; NOVÉ - JMÉNO (NEMĚL BY
CP      255              ; EXISTOVAT)
JP      NZ, CHYBA        ; POKUD EXISTUJE —> CHYBA
LD      C, 23            ; JINAK PŘEJMENUJ SOUBOR
LD      DE, FCB
CALL    BDOS
CP      255
JP      Z, CHYBA ; POKUD SE NEPOVEDLO —> CHYBA
; SOUBOR PŘEJMENOVÁN
```

Služby pro přístup k souborům

Modul BDOS poskytuje dva způsoby přístupu k vlastnímu obsahu souborů. Logické věty souboru je možno zpracovávat sekvenčně (služby 20 a 21), nebo používat přímý přístup k jednotlivým větám podle zadaného čísla věty (služby 33, 34, 36 a 40).

Přenos informace z disku do paměti nebo obráceně pracuje vždy s předem zadanou zónou paměti nazývanou zóna DMA. Její adresu lze nastavit voláním služby 26.

Služba 26: Nastavení adresy DMA

vstup: C – 1AH
DE – adresa zóny DMA
výstup: není

Služba 26 předpokládá adresu zóny DMA zadanou v registrovém páru DE. *Adresa zóny DMA* určuje počátek 128 slabik dlouhé vyrovnávací paměti pro diskové operace čtení a zápisu. Při spuštění programu v oblasti TPA je vždy nastavena na implicitní hodnotu 80H. Voláním služby 26 se adresa DMA nastaví na hodnotu uvedenou v registrovém páru DE. Nové nastavení platí až do dalšího volání služby 26, případně teplého nebo studeného startu nebo inicializace diskového systému (služba 13).

Služby pro sekvenční přístup

Služba 20: Sekvenční čtení věty

vstup: C – 14H
DE - adresa FCB
výstup: A – = 0 čtení proběhlo bez chyby
<> 0 chyba (čtení za koncem souboru)

Služba 20 realizuje *sekvenční čtení věty* ze souboru. Adresa řídicího bloku se zadává v registrech DE. Předpokládá se, že řídicí blok FCB byl aktivován službou 15 (otevření souboru). Služba 20 zajistí přečtení následující logické věty souboru do paměti, dle aktuální pozice v souboru. Aktuální pozice je určena aktuálním rozšířením (položka ex řídicího bloku) a číslem věty v aktuálním rozšíření (položkou cr v řídicím bloku).

Věta délky 128 slabik je zapsána do operační paměti na aktuální nastavenou adresu DMA (viz služba 26). Po přečtení věty je položka cr zvětšena o 1, aby pro příští čtení byla připravena následující věta. Pokud tím bylo aktuální rozšíření vyčerpáno, je automaticky otevřeno rozšíření následující a položka cr je vynulována. Služba vrací v registru A nulu, bylo-li čtení

úspěšné, v opačném případě (např. při čtení za koncem souboru) hodnotu od nuly různou. Následující program ilustruje použití služby pro sekvenční čtení ze souboru.

```
LD      C,26            ; NASTAV DMA NA BUF
LD      DE, BUF
CALL    BDOS
LD      A, 1            ; NASTAV ČÍSLO VĚTY
LD      (FCBCR), A
LD      C, 20           ; ČTI 1. VĚTU
LD      DE, FCB         ; V AKTUÁLNÍM ROZŠÍŘENÍ
CALL    BDOS
CP      0
JP      NZ, CHYBA       ; CHYBA PŘI ČTENÍ
; 1. VĚTA PŘEČTENA NA ADRESU BUF
```

Služba 21: Sekvenční zápis věty

vstup: C – 15H
DE – adresa FCB
výstup: A – = 0 úspěšný zápis
<> 0 chyba (zaplněný disk nebo adresář)

Sekvenční zápis věty do souboru umožňuje služba 21. Před zápisem do souboru je třeba aktivovat řídicí blok souboru FCB voláním služby 22 (vytvoř soubor), příp. služby 15 (otevři soubor), mají-li se věty zapisovat do existujícího souboru.

Voláním služby 21 je zapsána jedna logická věta do aktuálního nastaveného rozšíření souboru (položka ex řídicího bloku FCB), do pozice určené obsahem položky cr řídicího bloku FCB a položka cr je zvětšena o 1. Pokud takto byla zapsána poslední věta rozšíření (128. věta), je toto rozšíření automaticky uzavřeno, tj. zapsáno do adresáře na disk, a je vytvořeno rozšíření nové. Položka cr je v tom případě vynulována, položka ex zvětšena o 1.

Nenulová hodnota v registru A po návratu signalizuje chybu při zápisu, jejíž příčinou je obvykle přeplněný disk nebo adresář. Pokud aktuální věta v souboru již existovala, nový zápis staré věty přepíše. Použití ilustruje následující program.

```
LD      C, 26           ; NASTAV DMA NA BUF
LD      DE, BUF
CALL    BDOS
LD      A, 1            ; NASTAV ČÍSLO VĚTY
LD      (FCBCR), A
LD      C, 21           ; ZAPIŠ 1. VĚTU
LD      DE, FCB         ; V AKTUÁLNÍM ROZŠÍŘENÍ
CAL     BDOS
CP      0
JP      NZ, CHYBA       ; CHYBA PŘI ZÁPISU
; 1. VĚTA ZAPSÁNA
```

Služby pro přímý přístup

Služba 36: Výpočet čísla věty pro přímý přístup

vstup: C – 24H
DE – adresa FCB
výstup: položky r0, r1 v FCB

Služba 36 usnadňuje přechod ze sekvenčního zpracování na přímý přístup. Do položek r0 a r1 řídicího bloku FCB souboru zpracovávaného sekvenčně nastaví *číslo věty pro přímý přístup*. Číslo věty je spočteno na základě položek ex (aktuální rozšíření) a cr (číslo věty v aktuálním rozšíření). Takto lze sekvenčně procházet soubor a určovat čísla vět, ke kterým je potom možno přistupovat přímo.

Služba 33: Čtení věty s přímým přístupem

vstup: C – 21H
DE – adresa FCB
výstup: A – návratový kód:
= 0 úspěšná operace
1 čtení nezapsané věty
3 nelze uzavřít akutální rozšíření
4 čtení z nezapsaného rozšíření
6 příliš velké číslo věty

Pro *čtení věty s přímým přístupem* je určena služba 33. Služba 33 odpovídá službě 20 (sekvenční čtení věty) s tím rozdílem, že je přečtena věta, jejíž číslo je zadáno v položkách r0 a r1 řídicího bloku FCB (r0 nižší slabika čísla, r1 vyšší slabika čísla, r2 musí být nula). Číslo logické věty musí tedy být v rozsahu od 0 do 65 535.

Po návratu ze služby 33 obsahuje registr A návratový kód, který signalizuje úspěšnost operace čtení. Hodnota 0 signalizuje úspěšně provedenou operaci, nenulová hodnota popisuje chybu:

A = 1 pokus o čtení věty, která nebyla předtím do souboru zapsána (při přímém přístupu lze vytvářet soubory s "dírami"),

A = 3 systém nemůže změnit pozici v souboru dle požadavku, neboť nemůže uzavřít aktuální rozšíření; např. pokud aktuální rozšíření nebylo dosud vůbec vytvořeno,

A = 4 pokus o čtení z rozšíření, které dosud nebylo vytvořeno (podobně jako pro A=1),

A = 6 pokus o čtení logické věty, jejíž číslo je větší než 65 535, tj. položka r2 není nulová.

Aby byl soubor správně zpracován, musí být řídicí blok FCB aktivní (tj. musí předcházet některá služba, která jej aktivuje). Operace přímého a sekvenčního přístup k logickým větám souboru lze libovolně kombinovat. Po operaci s přímým přístupem lze pokračovat v přístupu sekvenčním s tím, že poslední věta přečtená, resp. zapsaná s přímým přístupem je přečtena, resp. zapsána ještě jednou. Číslo věty pro přímý přístup (položky r0 a r1), ani číslo věty pro sekvenční přístup (položka cr) se operací přímého přístupu nemění. Použití služby 33 ilustruje následující příklad.

```
LD    C, 26          ; NASTAV DMA NA BUF
LD    DE, BUF
CALL  BDOS
LD    A, 1           ; NASTAV ČÍSLO VĚTY
LD    (FCBCR), A     ; V ROZŠÍŘENÍ
LD    A, 1           ; NASTAV ČÍSLO ROZŠÍŘENÍ
LD    (FCBEX), A
LD    C, 36          ; NASTAV ČÍSLO VĚTY PRO
LD    DE, FCB        ; PŘÍMÝ PŘÍSTUP
```

```
LD      C, 33           ; ČTI VĚTU
LD      DE, FCB
CALL    BDOS
CP      0
JP      NZ, CHYBA       ; CHYBA PŘI ČTENÍ
; VĚTA PŘEČTENA
```

Služba 34: Zápis s přímým přístupem

vstup: C – 22H
DE – adresa FCB
výstup: A – návratový kód
= 0 úspěšná operace
3 nelze uzavřít aktuální rozšíření
5 přeplnění adresáře
6 příliš velké číslo věty

Zápis věty s přímým přístupem provádí služba 34. Při zápisu věty s přímým přístupem se zapíše jedna logická věta do souboru na pozici určenou číslem věty v položkách r0 a r1 řídicího bloku FCB. Jestliže příslušné rozšíření souboru doposud neexistuje, je automaticky vytvořeno.

Po návratu ze služby 34 je v registru A uveden návratový kód. Hodnota 0 signalizuje úspěšné provedení operace zápisu, nenulová hodnota signalizuje chybu:

A = 3 systém nemůže změnit pozici v souboru dle požadavku, neboť nemůže uzavřít akutální rozšíření; např. pokud aktuální rozšíření nebylo dosud vůbec vytvořeno,

A = 5 zápis věty do souboru vyžaduje vytvoření dalšího rozšíření, ale v adresáři disku již není žádná volná položka (adresář je plný),

A = 6 pokus o čtení logické věty, jejíž číslo je větší než 65 535, tj. položka r2 není nulová.

Má-li být přímým přístupem zapisováno do dosud neexistujícího souboru, musí být nejprve voláním služby 22 vytvořeno nulté rozšíření souboru. V opačném případě by mohl být soubor chybně zpracován.

Soubor vytvořený přímým přístupem může obsahovat "díry" (nezapsané věty). Při sekvenčním zpracovávání takovéhoto souboru může dojít

k předčasnému ohlášení konce souboru. Zápis do souboru s využitím přímého přístupu ilustruje následující program.

```
LD      C, 26               ; NASTAV DMA NA BUF
LD      DE, BUF
CALL    BDOS
LD      A, 1                ; NASTAV ČÍSLO VĚTY
LD      (FCBCR), A          ; V ROZŠÍŘENÍ
LD      A, 1        ;       NASTAV ČÍSLO ROZŠÍŘENÍ
LD      (FCBEX), A
LD      C, 36               ; NASTAV ČÍSLO VĚTY PRO
LD      DE, FCB             ; PŘÍMÝ PŘÍSTUP
LD      C, 34               ; ZAPIŠ VĚTU
LD      DE, FCB
CALL    BDOS
CP      0
JP      NZ, CHYBA           ; CHYBA PŘI ZÁPISU
; VĚTA ZAPSÁNA
```

Při vytváření souboru lze též použít službu 40 pro zápis věty s přímým přístupem s doplněním nul.

Služba 40: **Zápis věty s přímým přístupem s doplněním nul**

vstup: **C – 28H**
DE - adresa FCB
výstup: **A – návratový kód:**
= 0 úspěšná operace
3 nelze uzavřít aktuální rozšíření
5 přeplnění adresáře
6 příliš velké číslo věty

Služba 40 odpovídá službě 34 (zápis věty s přímým přístupem) s tím rozdílem, že při zápisu první věty do alokačního bloku jsou automaticky zapsány všechny zbývající věty alokačního bloku vyplněné binárními nulami. Službu lze použít např. pro vyplnění obsahu souboru pro přímý přístup binárními nulami.

Po návratu ze služby 40 je v registru A uveden návratový kód. Hodnota 0 signalizuje úspěšné provedení operace zápisu, nenulová hodnota signalizuje chybu:

A = 3 systém nemůže změnit pozici v souboru dle požadavku, neboť nemůže uzavřít aktuální rozšíření; např. pokud aktuální rozšíření nebylo dosud vůbec vytvořeno,

A = 5 zápis věty do souboru vyžaduje vytvoření dalšího rozšíření, ale v adresáři disku již není žádná volná položka (adresář je plný),

A = 6 pokus o čtení logické věty, jejíž číslo je větší než 65 535, tj. položka r3 není nulová.

Zjištění velikosti souboru

Služba 35: Výpočet velikosti souboru

vstup: C – 23H
DE – adresa FCB
výstup: položky r0, r1 v FCB – velikost souboru ve větách

Výpočet velikosti souboru provádí služba 35. Pro soubor určený jednoznačným jménem v řídicím bloku FCB služba 35 zjistí číslo poslední zapsané věty a číslo věty následující uloží do položek r0 a r1 řídicího bloku FCB. Jestliže je hodnota položky r2 různá od nuly, obsahuje soubor 65 536 vět.

Po vyvolání služby 35 lze připojovat data na konec souboru zápisem se sekvenčním (služba 21) i přímým (služba 34) přístupem.

Byl-li soubor vytvořen zápisem při sekvenčním přístupu, odpovídá velikost souboru v položkách r0 a r1 počtu zapsaných vět. V souboru vytvořeném přímým přístupem mohou být "díry".

Práce s atributy souboru

Služba 30: Nastavení atributů souboru

vstup: C – 1EH
DE – adresa FCB
výstup: A – adresářový kód

Služba 30 umožňuje *nastavení atributů souboru.* Vstupním parametrem je adresa řídicího bloku FCB s jednoznačným jménem souboru. Je-li

příslušný soubor nalezen v adresáři, jsou mu nastaveny atributy podle atributů uvedených v řídicím bloku FCB. Atributy jsou reprezentovány pomocí příznaků v identifikaci souboru, kde se využívá fakt, že znaky jsou zaznamenávány pouze v sedmibitovém kódu. Příznaky tvoří nejvyšší bity jména a typu souboru. Ve verzi 2.2 jsou definovány významy příznaků následovně:

Příznak	Atribut souboru
t1=1	R/O (Read Only) – zákaz zápisu do souboru,
t1=0	R/W (Read/Write) – soubor lze rušit a přepisovat,
t2=1	SYS (SYStem) – soubor se nevypisuje příkazem DIR,
t2=0	DIR (DIRectory) – soubor se vypisuje v adresáři,
t3=1	archivní bit – do souboru se zapisovalo,
t3=0	archivní bit – soubor nebyl modifikován od posledního nulování archivního bitu.

Příznaky f5 až f8 jsou rezervovány pro pozdější použití ve vyšších verzích, příznaky f1 až f4 jsou volné.

Ovládání aktuálního uživatele

Služba 32: Nastavení, příp. dodání čísla aktuálního uživatele

vstup: C – 20H
E – 255, příp. číslo uživatele
výstup: A – číslo aktuálního uživatele (pro E=255), jinak (E<>255) výstup není definován

Pomocí služby 32 modulu BDOS lze též ovládat *nastavení aktuálního uživatele*, příp. zjistit *číslo aktuálního uživatele*. Vstupním parametrem služby 32 je hodnota v registru E. Je-li v registru E hodnota 255, vrací služba 32 v registru A číslo aktuálního uživatele. Je-li zadána hodnota různá od 255, je číslo aktuálního uživatele změněno na tuto hodnotu (protože povolená čísla uživatelů jsou 0 až 15, je obsah registru E uvažován modulo 16).

5.5 Příklad programu

Pro ilustraci používání služeb modulu BDOS uvedeme příklad řešení modulu pro práci se souborem, který má věty libovolné pevné délky v rozsahu od 1 do 65 536 slabik. Modul využívá služeb modulu BDOS a realizuje služby pro přímý přístup k větám takto organizovaného souboru. Navíc je rozšířen o služby pro rušení a přejmenování souboru.

```
; MODUL PRO PŘÍMÝ PŘÍSTUP K SOUBORU S LOGICKÝMI
; VĚTAMI LIBOVOLNÉ DÉLKY.

; MODUL POSKYTUJE SLUŽBY PRO PRÁCI S JEDNÍM
; SOUBOREM, KTERÝ SE SKLÁDA Z VĚT LIBOVOLNÉ DÉLKY
; V ROZSAHU 1 ... 65 536 SLABIK. DÉLKA VĚTY JE ZADÁNA PŘI
; OTEVŘENÍ SOUBORU A PLATÍ PRO VŠECHNY NÁSLEDUJÍCÍ
; OPERACE ČTENÍ A ZÁPISU.
; POSKYTOVANÉ SLUŽBY:
;       OPEN ..... OTEVŘENÍ SOUBORU
;       CLOSE ..... UZAVŘENÍ SOUBORU
;       DELETE ..... RUŠENÍ SOUBORU (SOUBORŮ)
;       RENAME .... PŘEJMENOVÁNÍ SOUBORU
;       READ ..... ČTENÍ LOGICKÉ VĚTY
;       WRITE ..... ZÁPIS LOGICKÉ VĚTY
; POZNÁMKY: SLUŽBY OPEN, DELETE A RENAME LZE POUŽÍT
; POUZE NA SOUBOR (SOUBORY), KTERÉ NEBYLY OTEVŘENY
; SLUŽBOU OPEN. SLUŽBU CLOSE LZE POUŽÍT POUZE NA
; SOUBOR, KTERÝ BYL OTEVŘEM SLUŽBOU OPEN.

        .Z80

        PUBLIC OPEN
        PUBLIC CLOSE
        PUBLIC DELETE
        PUBLIC RENAME
        PUBLIC READ
        PUBLIC WRITE
```

```
; DEKLARACE
BDOS    EQU 5   ; VSTUPNÍ BOD PRO SLUŽBY BDOS

; SYMBOLICKÉ KÓDY SLUŽEB MODULU BDOS

FOPEN   EQU 15  ; OTEVŘENÍ SOUBORU
FCLOSE  EQU 16  ; UZAVŘENÍ SOUBORU
FFIRST  EQU 17  ; VYHLEDÁNÍ SOUBORU
FDELETE EQU 19  ; ZRUŠENÍ SOUBORU
FCREATE EQU 22  ; VYTVOŘENÍ SOUBORU
FRENAME EQU 23  ; PŘEJMENOVÁNÍ SOUBORU
FSETDMA EQU 26  ; NASTAVENÍ ADRESY DMA
FRREAD  EQU 33  ; ČTENÍ SEKTORU
FRWRITE EQU 34  ; ZÁPIS SEKTORU
FRAND   EQU 36  ; VÝPOČET ČÍSLA SEKTORU PRO PŘÍMÝ
                ; PŘÍSTUP

; VNITŘNÍ PROMĚNNÉ MODULU PRO OPERACE SE SOUBOREM

        DSEG

SIZE:   DW  0   ; DÉLKA LOGICKÉ VĚTY
FCB:    DS  36  ; ŘÍDICÍ BLOK SOUBORU
BUF:    DS  128 ; VYROVNÁVACÍ PAMĚŤ NA SEKTOR
REST:   DW  0   ; ZBÝVÁ PŘENÉST (VE SLABIKÁCH)
FCB2:   DS  54  ; POMOCNÉ FCB PRO SLUŽBY RUŠENÍ A
                ; PŘEJMENOVÁNÍ SOUBORU. DÉLKA JE
                ; VĚTŠÍ (JEDEN A PŮL FCB) KVŮLI
                ; SNADNÉ KONTROLE.

; POLOŽKY FCB

FCBDR   EQU FCB     ; OZNAČENÍ DISKOVÉ JEDNOTKY
FCBNAME EQU FCB+01  ; JMÉNO SOUBORU
FCBTYPE EQU FCB+09  ; TYP SOUBORU
FCBEX   EQU FCB+12  ; ČISLO AKTUÁLNÍHO ROZŠÍŘENÍ
FCBS2   EQU FCB+14  ;
```

```
FCBDR2   EQU  FCB+16      ; JEDNOTKA PRO RENAME
FCBNAM2  EQU  FCB+17      ; JMÉNO PRO RENAME
FCBR0    EQU  FCB+33      ; ČÍSLO SEKTORU PRO PŘÍMÝ
FCBR2    EQU  FCB+35      ; PŘÍSTUP

         CSEG

;=======================================
; POMOCNÉ PODPROGRAMY
;=======================================

; PODPROGRAM MOVNAM:
;----------------------------------------

; PODPROGRAM MOVNAM SLOUŽÍ PRO PŘESUN JMÉNA
; SOUBORU DO FCB.
; JMÉNO MUSÍ BÝT ULOŽENO NA ADRESE ZADANÉ V REGIS -
; TROVÉM PÁRU DE VE TVARU (DÉLKA JE PŘESNĚ 14 SLABIK):
; D: NNNNNNNN.TTT, KDE:
;      D ..... DISKOVÁ JEDNOTKA,
;      NNNNNNNN .... JMÉNO SOUBORU A
;      TTT ..... ..TYP SOUBORU.
; JMÉNO A TYP SOUBORU MUSÍ BÝT DOPLNĚNO NA
; PŘÍSLUŠNOU DÉLKU MEZERAMI.
; VSTUP    DE .. ADRESA JMÉNA SOUBORU
;          HL .. ADRESA ŘÍDÍCÍHO BLOKU SOUBORU
; VÝSTUP:  -
; VOLÁNÍ:
;          LD   DE,NAME    ; ADRESA JMÉNA SOUBORU
;          LD   HL,FCB     ; ADRESA ŘÍDICÍHO BLOKU
;          CALL MOVNAM

MOVNAM:  PUSH BC           ; USCHOVEJ REGISTRY BC.
         LD   A, (DE)      ; PŘESUN JMÉNA SOUBORU DO
                           ; FCB.
```

```
        SUB   'B'          ; KÓDOVÁNÍ JEDNOTEK JE:
                           ; 0 = AKTUÁLNÍ, 1 = A, 2 = B, ATD.
        LD    (HL), A      ; NASTAV JEDNOTKU V FCB
        INC   DE           ; PŘESKOK JEDNOTKY
        INC   DE           ; PŘESKOK " : "
        EX    DE,HL
        INC   DE
        LD    BC,8
        LDIR               ; PŘESUN JMÉNA DO FCB
        INC   HL           ; PŘESKOK " . "
        LD    BC,3
        LDIR               ; PŘESUN TYPU DO FCB
        POP   BC           ; OBNOV REGISTRY BC
        RET

; PODPROGRAM FNUMB
; ---------------------------------
; PODPROGRAM FNUMB SLOUŽÍ PRO VÝPOČET DISKOVÉ
; ADRESY ZADANÉ LOGICKÉ VĚTY.
; VSTUP:  HL . . LOGICKÉ ČÍSLO VĚTY (1 ... 65 536)
; VÝSTUP: HL . . LOGICKÉ ČÍSLO SEKTORU CPM
;         (SEKTOR 0 ... 65 355)
;         DE . . POSUN ZAČÁTKU LOGICKÉ VĚTY V SEKTORU
;         (0 ... 127)

FNUMB:  PUSH BC            ; ÚSCHOVA REGISTRŮ BC
        EX   DE, HL        ; VÝPOČET A, HL: = HL*SIZE
        LD   HL,0
        LD   BC, (SIZE)
        LD   A,0
        SCF
        CCF
MULT:   DEC  DE
        PUSH AF
        LD   A, E
        JR   NZ, MULT1
```

```
            LD    A,D
            CP    0
            JP    Z, N1
MULT1:      POP   AF
            ADD   HL, BC
            JR    NC, MULT
            INC   A
            CCF
            JR    MULT
; ZDE TROJICE REGISTRŮ A, H, L OBSAHUJE VÝSLEDNÝ POSUN
; ZAČÁTKU LOGICKÉ VĚTY OD POČÁTKU SOUBORU VE
; SLABIKÁCH
N1:         POP   AF
            PUSH  HL          ; SEKTOR :   HL : = A.HL DIV 128
            POP   DE          ; POSUV:     DE : = A,HL MOD 128
            LD    D,0
            SLA   E
            SRL   E
            LD    B,7
N2:         SRL   A
            RR    H
            RR    L
            DJNZ  N2
            POP   BC          ; OBNOVA REGISTRŮ BC
            RET

; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
; PODPROGRAM OPEN: OTEVŘENÍ SOUBORU
; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
; VSTUP:    BC . .  DÉLKA VĚTY PRO ČTENÍ A ZÁPIS
;           DE . .  ADRESA IDENTIFIKACE SOUBORU
; VÝSTUP :  A = 255 . . .  SOUBOR SE NEPODAŘILO OTEVŘÍT
;           (SOUBOR NELZE OTEVŘÍT ANI VYTVOŘIT)
;           A < > 255 . . ADRESÁŘOVÝ KÓD
; VOLÁNÍ:
;           LD    BC,DELKA  ; DÉLKA VĚTY
```

```
;           LD    DE, NAME  ; ADRESA JMÉNA SOUBORU
;           CALL  OPEN

OPEN:       ; PŘESUN DÉLKY DO PROMĚNNÉ SIZE
            LD    (SIZE), BC
            ; INICIALIZACE ŘÍDÍCÍHO BLOKU FCB
            LD    HL, FCB ; PŘESUN IDENTIFIKACE SOUBORU
            CALL  MOVNAM
            LD    A,0                ; DALŠÍ INICIALIZACE
            LD    (FCB+ FCBEX), A    ; NASTAV EX
            LD    (FCB+FCBS2), A     ; NASTAV S2
            LD    C, FIRST
            LD    DE,FCB      ; HLEDEJ, ZDA SOUBOR EXISTUJE
            CALL  BDOS
            CP    255
            JP    Z, CREATE   ; POKUD NEEXISTUJE —>
                              ; VYTVOŘ JEJ
            LD    C,FOPEN     ; JINAK OTEVŘI EXISTUJÍCÍ
                              ; SOUBOR
            LD    DE, FCB
            CALL  BDOS
            RET
CREATE:     LD    DE, FCB     ; VYTVOŘENÍ NOVÉHO SOUBORU
            LD    C,FCREATE
            CALL  BDOS
            RET

;=================================
; PODPROGRAM CLOSE: UZAVŘENÍ SOUBORU
;=================================
; VSTUP:    -
; VÝSTUP:   A = 255 . . .  CHYBA - SOUBOR NELZE UZAVŘÍT
;           A <> 255 . . ADRESÁŘOVÝ KÓD
; VOLÁNÍ:
;           CALL  CLOSE
```

```
CLOSE:  LD   C,FCLOSE
        LD   DE,FCB
        CALL BDOS
        RET

;=================================
; PODPROGRAM DELETE: ZRUŠENÍ SOUBORU
;=================================
; VSTUP:  DE.....ADRESA IDENTIFIKACE SOUBORU
; VÝSTUP: A = 255... CHYBA - SOUBOR NELZE ZRUŠIT
;         A <> 255.. ADRESÁŘOVÝ KÓD
; VOLÁNÍ:
;         LD   DE, NAME  ; ADRESA JMÉNA SOUBORU
;         CALL DELETE

DELETE: PUSH BC          ; ÚSCHOVA REGISTRŮ BC
        PUSH HL                  ; ÚSCHOVA RAGISTRŮ HL
        LD   HL,FCB2     ; PŘESUN IDENTIFIKACE DO FCB2
        CALL MOVNAM
        LD   A,0                 ; DALŠÍ INICIALIZACE
        LD   (FCB2+FCBEX),A  ; NASTAV EX
        LD   (FCB2+FCBS2),A  ; NASTAV S2
        LD   C, FFIRST
        LD   DE, FCB2    ; HLEDEJ, ZDA SOUBOR EXISTUJE
        CALL BDOS
        CP   255
        JP   Z, CHYBAD ; POKUD NEEXISTUJE —> CHYBA
        LD   C,FDELETE ; JINAK ZRUŠ SOUBOR
        LD   DE,FCB2
        CALL BDOS
DELOK:  POP  HL                  ; OBNOVA REGISTRŮ HL
        POP  BC          ; OBNOVA REGISTRŮ BC
        RET
CHYBAD:                  ; BEZ HLÁŠENÍ CHYB
        JP   DELOK
```

```
; ========================================
; PODPROGRAM RENAME: PŘEJMENOVÁNÍ SOUBORU
; ========================================
; VSTUP:  HL..  NOVÉ-JMÉNO: ADRESA ZÓNY ZNAKŮ
;               DÉLKY 14 VE TVARU D: NNNNNNNN . TTT
;         DE..  STARÉ-JMÉNO: ADRESA ZÓNY ZNAKŮ
;               DÉLKY 14 VE TVARU D: NNNNNNNN . TTT
; VÝSTUP: A.....  STAV PROVEDENÉ OPERACE
;         A = 0..  BEZ CHYBY
;         A < > 0  PŘI PROVÁDĚNÍ OPERACE NASTALA
;                  CHYBA
;         A = 1..  SOUBOR NOVÉ-JMÉNO JIŽ EXISTUJE
;         A = 2..  SOUBOR STARÉ-JMÉNO NEEXISTUJE

RENAME: PUSH BC          ; ÚSCHOVA REGISTRŮ BC
        PUSH HL          ; SCHOVEJ ADRESU NOVÉ-JMÉNO
        LD   HL, FCB2    ; PŘESUŇ STARÉ-JMÉNO DO FCB2
        CALL MOVNAM
        POP  DE          ; PŘESUŇ NOVÉ-JMÉNO DO
                         ; FCB2+FCBDR2
        LD   HL, FCB2+FCBDR2
        LD   A,0                 ; DALŠÍ INICIALIZACE
        LD   (FCB2+FCBEX),A      ; NASTAV EX
        LD   (FCB2+FCBEX),A      ; NASTAV S2
        LD   C, FOPEN
        LD   DE, FCB2+FCBDR2 ; POKUSNĚ OTEVŘI
        CALL BDOS                ; SOUBOR NOVÉ-JMÉNO
        CP   255                 ; (NEMĚL BY EXISTOVAT)
        LD   A,1                 ; NASTAV A - NEZMĚNÍ
PŘÍZNAKY
        JP   NZ,CHYBA  ; POKUD EXISTUJE —> CHYBA
        LD   C, FRENAME      ; JINAK PŘEJMENUJ SOUBOR
        LD   DE,FCB2
        CALL BDOS
        CP   255
        LD   A,2         ; NASTAV A - NEZMĚNÍ PŘÍZNAKY
```

```
            JP      Z, CHYBA    ; POKUD SE NEPOVEDLO —>
                                ; CHYBA
            LD      A,0         ; JINAK BEZ CHYBY
CHYBA:
RENOK:
RENRET:     POP     BC          ; OBNOVA REGISTR BC
            RET

;=================================
; PODPROGRAM READ: ČTENÍ LOGICKÉ VĚTY
;=================================
; VSTUP:    HL..  ADRESA PRO ČTENÍ VĚTY (DÉLKA JE V SIZE)
;           DE..  LOGICKÉ ČÍSLO ČTENÉ VĚTY (1 ... 65 536)
; VÝSTUP:   A.....  STAV PROVEDENÉ OPERACE
;           A = 0..  BEZ CHYBY
;           A < > 0  PŘI ČTENÍ NASTALA CHYBA
;                    (CHYBOVÝ KÓD VIZ POPIS FUNKCÍ 33 A 34)

READ:       PUSH BC                 ; ÚSCHOVA REGISTRŮ BC
            PUSH HL                 ; ÚSCHOVA ADRESY
            EX   DE,HL              ; HL: LOGICKÉ ČÍSLO VĚTY
            CALL FNUMB              ; SPOČTENÍ DISKOVÉ
                                    ; ADRESY VĚTY
            LD   (FCB+FCBR0),HL     ; ULOŽ ČÍSLO SEKTORU
                                    ; DO FCB
            LD   A,O
            LD   (FCB+FCBR2), SA    ; NASTAV R2
            PUSH DE                 ; SCHOVEJ POSUN
            LD   C, FSETDMA         ; NASTAV DMA NA BUF
            LD   DE, BUF
            CALL BDOS
            LD   HL, (SIZE) ; REST:=SIZE (ZBÝVÁ PŘESUNOUT)
            LD   (REST), HL
RD0:
            LD   HL,REST            ; IF REST=0
            LD   A, (HL)
```

```
        CP    0
        JR    NZ,RD01
        INC   HL
        LD    A, (HL)
        CP    0
        JP    Z, RDOK     ; THEN KONEC
RD01:   LD    C, FRREAD   ; ELSE ČTI SEKTOR
        LD    DE,FCB
        CALL  BDOS
        CP    0
        JP    NZ, CHYBAR; CHYBA PŘI ČTENÍ SEKTORU
        LD    HL, BUF     ; HL:=BUF+POSUN
        POP   DE
        ADD   HL, DE
        PUSH  HL
        LD    A, 128      ; DÉLKA:=BUF+128 - HL
        SUB   E
        LD    C, A
        LD    B, 0
        LD    HL, (REST)
        SCF
        CCF
        SBC   HL, BC
        JP    P, RD1      ; IF (REST > DÉLKA) THEN —> RD1
        LD    HL, (REST)          ; ELSE (REST < = DÉLKA)
        LD    B, H                ; DÉLKA:=REST
        LD    C,L
        LD    HL, 0               ; REST:=0
RD1:    LD    (REST), HL          ; REST:=REST - DÉLKA
        POP   HL                  ; HL - ODKUD
        POP   DE                  ; DE - KAM
        PUSH  DE                  ; BC - DÉLKA
        PUSH  BC
        LDIR                      ; PŘESUN ČÁSTI VĚTY
        POP   BC
        POP   HL
```

```
            ADD   HL, BC
            PUSH  HL                    ; KAM:=KAM+DÉLKA

            LD    HL, 0
            PUSH  HL                    ; POSUN:=0
            LD    HL, (FCB+FCBR0)       ; DALŠÍ SEKTOR V FCB
            INC   HL
            LD    (FCB+FCBR0), HL
            JP    RDO
CHYBAR:     ; HLÁŠENÍ CHYB
            JP    RD3
RDOK:       LD    A,0
RD3:        POP   HL          ; UPRAV ZÁSOBNÍK A NÁVRAT
            POP   HL
            POP   BC          ; OBNOVA REGISTRŮ BC
            RET

;=================================
; PODPROGRAM WRITE: ZÁPIS LOGICKÉ VĚTY
;=================================
; VSTUP:  HL.. ADRESA ZAPISOVANÉ VĚTY
;              (DÉLKA JE V SIZE)
;         DE.. LOGICKÉ ČÍSLO ZAPISOVANÉ VĚTY (1...65 536)
; VÝSTUP: A..... STAV PROVEDENÉ OPERACE
;         A = 0 .. BEZ CHYBY
;         A < > 0  PŘI ZÁPISU NASTALA CHYBA
;                  (CHYBOVÝ KÓD VIZ POPIS FUNKCÍ 33 A 34)

WRITE:      PUSH  BC          ; ÚSCHOVA REGISTRŮ BC
            PUSH  HL          ; ÚSCHOVA ADRESY VĚTY
            EX    DE, HL      ; HL:= LOGICKÉ ČÍSLO VĚTY
            CALL  FNUMB       ; SPOČTENÍ DISKOVÉ ADRESY
                              ; VĚTY
            LD    (FCB+FCBR0), HL  ; ULOŽ ČÍSLO SEKTORU
                                   ; DO FCB
```

```
        LD    A,0
        LD    (FCB+FCBR2),A     ; NASTAV R2
        PUSH  DE                ; SCHOVEJ POSUN
        LD    (FCB+FCBR2), A    ; NASTAV DMA NA BUF
        LD    DE, BUF
        CALL  BDOS
        LD    HLK, (SIZE)       ; REST:=SIZE (ZBÝVÁ
                                ; PŘESUNOUT)
        LD    (REST), HL
WR0:
        LD    HL, REST    ; IF REST=0
        LD    A, (HL)
        LD    B, A
        INC   HL
        LD    A, (HL)
        OR    B
        CP    0
        JP    Z, WROK     ; THEN KONEC
WR01:   LD    C,FRREAD    ; ELSE ČTI SEKTOR
        LD    DE, FCB
        CALL  BDOS
        CP    0
        JR    Z,WR011     ; ČTENÍ BEZ CHYBY —> WR011
        CP    1     ; ČTENÍ NEZAPSANÉHO SEKTORU DAT
        JP    Z,WR010                ;       —> WR010
        CP    4     ; ČTENÍ NEZAPSANÉHO ROZŠÍŘENÍ
                                     ;       —> WR010
        JP    NZ,CHYBAW ; JINAK CHYBA PŘI ČTENÍ
                        ; SEKTORU
WR010:  LD    HL, BUF     ; VYPLŇ BUF ZNAKEM EOF (1AH)
        LD    A,1AH
        LD    (HL), A
        LD    HL, BUF
        LD    DE, BUF+1
        LD    BC,127
        LDIR
```

```
WR011:  LD    HL, BUF     ; HL:=BUF+POSUN
        POP   DE
        ADD   HL, DE
        PUSH  HL
        LD    A, 128      ; DÉLKA:= BUF+128 - HL
        SUB   E
        LD    C, A
        LD    B,0
        LD    HL, (REST)
        SCF
        CCF
        SBC   HL, BC
        JP    P, WR3      ; IF (REST > DÉLKA) THEN —> WR3
        LD    HL, (REST)  ; ELSE (REST <= DÉLKA)
        LD    B, H        ; DÉLKA:=REST
        LD    C, L
        LD    HL, 0       ; REST=0
WR3:    LD    (REST), HL  ; REST:=REST - DÉLKA
        POP   DE          ; DE - KAM
        POP   HL          ; HL - ODKUD
        PUSH  HL          ; BC - DÉLKA
        PUSH  BC
        LDIR              ; PŘESUN ČÁSTI VĚTY
        POP   BC
        ADD   HL,BC
        PUSH  HL          ; ODKUD:=ODKUD+DÉLKA
        LD    HL,0
        PUSH  HL          ; POSUN:=0
        LD    C,FRWRITE   ; ZAPIŠ SEKTOR
        LD    DE, FCB
        CALL  BDOS
        CP    0
        JP    NZ, CHYBAW
```

```
        LD    HL, (FCB+FCBR0)   ; DALŠÍ SEKTOR V FCB
        INC   HL
        LD    (FCB+FCBR0), HL
        JP    WR0
CHYBAW  ;     HLÁŠENÍ CHYB
        JP    WRRET
WROK:   LD    A, 0              ; BEZ CHYBY
WRRET:  POP   HL                ; UPRAV ZÁSOBNÍK A NÁVRAT
        POP   HL
        POP   BC                ; OBNOVA REGISTRŮ BC
        RET

        END
```

6 Vazba systému na technické prostředky

Jak jsme se již zmínili v předcházejících kapitolách, skládá se operační systém CP/M z technicky závislých a technicky nezávislých částí. V této kapitole se budeme věnovat popisu technicky závislých částí systému CP/M, tj. popisu správy technických prostředků (modulu BIOS) a systémového zavaděče.

Technicky závislé části operačního systému CP/M vytvářejí programové prostředí pro práci ostatních složek programového vybavení. Z hlediska hierarchické architektury systému spolupracuje s technicky závislou vrstvou výhradně jádro systému (modul BDOS), jehož služeb pak využívají všechny vyšší složky architektury, tj. ostatní systémové i aplikační programy, včetně monitoru CCP.

Běžný programátor by měl využívat výhradně služeb jádra (modulu BDOS) a technicky závislé složky systému považovat za neviditelné. Tato kapitola je proto určena zejména pro takové uživatele systému CP/M, kteří chtějí systém upravit pro nové prostředí, či jinak zasahovat do technicky závislých částí. Mimo vlastní popis vazby systému na technické prostředky jsou zde uvedeny informace, potřebné při úpravách či vytváření technicky závislých složek systému.

6.1 Správa technických prostředků (modul BIOS)

Vazbu operačnícho systému CP/M na technické prostředky počítače zajišťuje modul BIOS. Hiearchicky vyšší složky systému (zejména modul BDOS, ale i modul CCP a ostatní programy) jsou na technickém vybavení

nezávislé. Spolupráci vyšších složek architektury (nadále jen modulu BDOS) s modulem BIOS umožňuje pevně stanovené rozhraní mezi moduly BIOS a BDOS. Toto rozhraní musí každá implementace modulu BIOS dodržet, jinak je tělo modulu BIOS závislé pouze na konkrétních technických prostředcích počítače.

Definice rozhraní mezi moduly BIOS a BDOS zahrnuje:
- definici umístění modulu BIOS,
- definici způsobu vyvolávání služeb modulu BIOS,
- repertoár služeb modulu BIOS a
- definici struktury parametrů, předávaných mezi moduly BDOS a BIOS.

V paměti je modul BIOS obvykle uložen bezprostředně za modulem BDOS. Přesná definice rozhraní však stanoví, že modul BIOS musí být uložen v operační paměti na adrese dělitelné 256 (od počátku stránky) a adresa stránky je uložena na absolutní adrese 2. Adresa stránky, kde je uložen modul BIOS, je součástí instrukce JP BIOS+3 (JMP BIOS+3) povinně uložené na adrese 0 v komunikační zóně. Systém dále předpokládá, že modul BIOS není uložen v operační paměti před modulem BDOS, neboť oblast do počátku modulu BDOS považuje systém za volnou paměť pro programy.

Z technického hlediska se modul BIOS skládá ze sady ovladačů přídavných zařízení. Systém CP/M rozeznává přídavná zařízení dvou typů – *znaková zařízení* a *diskové jednotky*. Je dána pevná sada služeb (funkcí), jejichž realizaci musí modul BIOS zajistit. Služby modulu BIOS lze rozdělit do tří kategorií:
- služby pro inicializaci systému,
- služby pro práci se znakovými zařízeními,
- služby pro práci s diskovými jednotkami.

Modul BIOS může zahrnovat ovladače až 16 znakových zařízení a až 16 diskových jednotek. Aby repertoár služeb nebyl příliš rozsáhlý, jsou stejné služby pro podobná zařízení sloučeny do služby jediné. Konkrétní zařízení je pak určeno jiným způsobem – v případě diskových jednotek nejblíže předchozím voláním služby "nastavení diskové jednotky", v případě znakových zařízení obsahem slabiky na adrese 3 (tzv. IOBYTE). Skupina podobných zařízení je označována jako *logické zařízení*.

Podobně jako repertoár služeb, je pevně stanoven i způsob jejich používání. Vazba na služby modulu BIOS je zajištěna skokovým vektorem na počátku modulu BIOS, tj. na adrese, jejíž stránka je uvedena na počátku

operační paměti (adresa 2). Skokový vektor tvoří sada 17 skoků do 17 standardních podprogramů modulu BIOS, uvedených v následujícím přehledu.

Vektor služeb modulu BIOS

Relativní adresa	Instrukce	Služba
00H	JP BOOT	studený start systému
03H	JP WBOOT	teplý start systému
06H	JP CONST	zjištění stavu konzole
09H	JP CONIN	vstup znaku z konzole
0CH	JP CONOUT	výstup znaku na konzoli
OFH	JP LIST	výstup znaku na tiskárnu
12H	JP PUNCH	výstup znaku na děrovač
15H	JP READER	vstup znaku ze snímače
18H	JP HOME	nastavení na stopu 0
1BH	JP SELDSK	nastavení diskové jednotky
1EH	JP SETTRK	nastavení stopy
21H	JP SETSEK	nastavení sektoru
24H	JP SETOMA	nastavení adresy DMA
27H	JP READ	čtení sektoru
2AH	JP WRITE	zápis sektoru
2DH	JP LISTST	zjištění stavu tiskárny
30H	JP SECTRAN	překlad čísel sektorů

Služby BOOT a WBOOT slouží pro inicializaci systému. Služby CONST, CONIN, CONOUT, LIST, PUNCH, READER a LISTST slouží pro obsluhu logických znakových zařízení (CON:, RDR:, PUN: a LST:). Ostatní služby HOME, SELDSK, SETTRK, SETSEK, SETDMA, READ, WRITE a SECTRAN jsou určeny pro obsluhu diskových jednotek.

6.1.1 Služby pro inicializaci systému

Služby pro inicializaci systému zahrnují především:
- kód pro nastavení (naprogramování) stykových obvodů, tj. inicializační části všech ovladačů.
- kód pro vytvoření obrazu rezidentní části systému v operační paměti.

Nastavení stykových obvodů je obvykle nutno provést pouze jednou – po zapnutí počítače či havárii systému. Proto je inicializace systému rozdělena na službu pro *prvé zavedení systému* (studený start – BOOT), kdy je nutno programovat stykové obvody a službu pro *restart systému* (teplý start – WBOOT), kdy je pouze vytvářen obraz systému v paměti.

BOOT – Studený start systému (Cold Start)

Tento vstupní bod není při běžné činnosti systému využíván. Řízení se na tuto adresu předává pouze po skončení činnosti systémového zavaděče (např. po zapnutí počítače). Před aktivací služby BOOT musí být obraz modulu BIOS (nebo minimálně kód služby BOOT) k dispozici v operační paměti.

Úkolem služby BOOT je *úplná inicializace* technického prostředí počítače a *vytvoření správného obrazu* systému v operační paměti. Služba BOOT musí provést všechny inicializační akce, které neprovedl technický ani systémový zavaděč, tj. nastavení všech dosud neinicializovaných stykových obvodů. Dále provádí inicializaci tabulek popisu diskových jednotek (DPH), které musí být uloženy v paměti typu RWM. Tato paměť musí být umístěna za modulem BIOS na vyšších adresách a slouží rovněž pro umístění proměnných modulu BIOS.

Systémový zavaděč, který aktivuje službu BOOT, obvykle vytváří obraz systému v operační paměti sám. Zejména to platí o obrazu modulů CCP a BDOS, které jsou standardně uloženy v dohodnuté systémové oblasti na disku. Vytvoření obrazu modulu BIOS může být realizováno různě (totéž ovšem platí i pro zavádění modulů CCP a BDOS, neboť systémový zavaděč patří k variabilním složkám systému).

Způsobů, kterými lze vytvořit obraz modulu BIOS v operační paměti, je více. V nejjednodušším případě je modul BIOS uložen v pevné paměti (ROM) přímo na těch adresách, na kterých se také provádí. V tomto případě provádí služba BOOT pouze inicializaci tabulek popisu diskových jednotek (DPH).

Je-li modul BIOS uložen v pevné paměti na jiné adrese než je jeho pracovní oblast, musí se při zavádění překopírovat na příslušné adresy paměti typu RWM. Po překopírování se může pevná paměť odpojit (lze využít

stínovou paměť). Kopírování i odpojení může být zahrnuto v systémovém zavaděči nebo jako součást kódu služby BOOT modulu BIOS.

Modul BIOS může být rovněž načten z disku. V tomto případě zajišťuje zavedení systémový zavaděč, který přenese obraz systému (nebo minimálně část obsahující kód služby BOOT) z dohodnuté oblasti na disku do operační paměti. Poté předá řízení službě BOOT, která může případně zajistit přenos zbytku obrazu systému.

V rámci studeného startu se dále provádí:

- inicializace technického vybavení (pokud nebyla provedena technickým nebo systémovým zavaděčem),
- nastavení instrukcí skoku v komunikační zóně, tj. uložení instrukce JP BIOS+3 na adresu 0 a instrukce JP BDOS+6 na adresu 5,
- nastavení počáteční hodnoty slabiky IOBYTE (adresa 3), pokud je v systému implementováno přiřazování logických a fyzických zařízení,
- nulování slabiky CDISK (adresa 4); tím je zajištěno, že po studeném startu se stane aktuální jednotka A: a uživatel 0,
- vypsání identifikační zprávy, která má obsahovat jméno a verzi operačního systému a velikost operační paměti, pro níž je systém nakonfigurován.

Další činnost spočívá v aktivaci monitoru CCP. Do registru C se uloží hodnota 0, neboť obsah registru C využívá monitor pro nastavení aktuální diskové jednotky. Adresa pro aktivaci monitoru se obvykle spočítá tak, že od počáteční adresy modulu BIOS se odečte konstanta 1600H (délka modulů CCP a BDOS) – předpokládá se, že moduly BDOS a CCP jsou uloženy bezprostředně před modulem BIOS. Monitor CCP obsahuje rovněž na počátku krátký skokový vektor – při studeném startu je obvykle aktivován vstupní bod CCP nebo CCP+3.

WBOOT – Teplý start systému (Warm Stat)

Na vstupní bod pro *teplý start systému* (BIOS+3 – označovaný jako WBOOT) směřuje skok z adresy 0 v komunikační zóně SPA. Skokem na adresu 0 (a tedy teplým startem) obvykle končí běh programu v zóně TPA.

V rámci teplého startu se nejprve zavedou do operační paměti obrazy modulů CCP a BDOS. Moduly se nejčastěji zavádějí z vymezené oblasti média v systémové jednotce A:. Například na standardní osmipalcové disketě s jednoduchou hustotou (formát IBM 3740) bývá systém uložen takto:

1. sektor 0. stopy – rezervováno pro systémový zavaděč (boot blok)
2. sektor 0. stopy až 17. sektor 0. stopy – modul CCP
18. sektor 0. stopy až 19. sektor 1. stopy – modul BDOS
20. sektor 1. stopy až 26. sektor 1. stopy – modul BIOS

Moduly CCP a BDOS mohou být rovněž zavedeny z pamětí typu ROM (originální systém CP/M nemůže být v paměti ROM prováděn, neboť v sobě obsahuje proměnné, které jsou modifikovány).

Po zavedení modulů CCP a BDOS následuje doplnění instrukcí skoku do komunikační zóny – instrukce JP WBOOT (JMP WBOOT) na adresu 0, kde WBOOT=BIOS–0E00H. Dále se nastaví implicitní adresa pro DMA na adresu 80H (do zóny SPA). Nakonec se řízení předá modulu CCP (skokem na adresu BIOS–1600H). V registru C se předává obsah slabiky CDISK (adresa 4) – číslo jednotky a uživatele, které se stanou aktuální jednotkou a aktuálním uživatelem.

6.1.2 Služby pro obsluhu znakových zařízení

Modul BIOS obsahuje vstupní body pro obsluhu čtyř logických znakových zařízení:
– systémové konzole (CON:),
– logického zařízení pro znakový vstup (RDR:),
– logického zařízení pro znakový výstup (PUN:),
– logického zařízení pro tisk (LST:):

Pro činnost systému je nezbytná pouze systémová konzole. Zařízení pro výstup tisku využívá systém pro kopii výstupu na systémovou konzoli (hardcopy – ^P), zařízení znakového vstupu a výstupu nevyužívá vůbec. Plné využití znakových periférií umožňuje služební program PIP.

Pro jednotlivá logická znaková zařízení jsou k dispozici operace:
– test stavu (CON:, LST:),
– vstup znaku (CON:, RDR:),
– výstup znaku (CON:, PUN:, LST:).

V rámci modulu BIOS může být implementována možnost přiřazení různých fyzických zařízení odpovídajícím zařízením logickým. Aktuální přiřazení je určeno obsahem slabiky IOBYTE (adresa 3), kde na každé logické zařízení jsou rezervovány 2 bity, tj. každému logickému zařízení lze přiřadit

až 4 různá fyzická zařízení (tab. 6.1). Na základě obsahu slabiky IOBYTE pak modul BIOS může směrovat stejnou službu na různá fyzická zařízení. Hodnota každé dvojice bitů slabiky IOBYTE určuje přiřazení podle tab. 6.1.

tab. 6.1.

Log. zař.	Bity IOBYTE	Bin. hod.	Fyzické zařízení
CON:	0,1	00	dálnopis (TTY:)
		01	obrazovkový displej (CRT:)
		10	dávkové zpracování (BAT: = vstup z PUN:, výstup na LST:)
		11	uživatelsky def. konzole (UC1:)
RDR:	2,3	00	dálnopis (TTY:)
		01	snímač děrné pásky (PTR:)
		10	1. uživatelsky definované zařízení znakového vstupu (UR1:)
		11	2. uživatelsky definované zařízení znakového vstupu (UR2:)
PUN:	4,5	00	dálnopis (TTY:)
		01	děrovač děrné pásky (PTP:)
		10	1. uživatelsky definované zařízení znakového výstupu (UP1:)
		11	2. uživatelsky definované zařízení znakového výstupu (UP2:)
LST:	6,7	00	dálnopis (TTY:)
		01	obrazovkový displej (CRT:)
		10	tiskárna (LPT:)
		11	uživatelsky definované zařízení výstupu tisku (UL1:)

CONST – Test stavu systémové konzole (CONsole STatus)

Služba CONST vrací v registru A hodnotu 255, byla-li na systémové konzoli stisknuta klávesa (je připraven znak ke čtení). V opačném případě vrací v registru A hodnotu 0.

CONIN – Čtení znaku ze systémové konzole (CONsole INput)

Služba CONIN čeká na stisk klávesy na systémové konzoli a vrací v registru A kód stisknutého znaku s nulovým paritním bitem (bit číslo 7). Služba CONIN neprovádí automaticky kontrolní opis znaku na výstup systémové konzole (echo).

CONOUT – Výstup znaku na systémovou konzoli (CONsole OUTput)

Služba CONOUT zajistí výstup znaku z registru C na systémovou konzoli.

LIST – Výstup znaku z registru C na logické zařízení pro výstup tisku
PUNCH – Výstup znaku z registru C na logické zařízení pro znakový výstup
READER – Čtení znaku z log. zařízení pro znakový vstup do registru A

LISTST – Test připravenosti logického zařízení výstupu tisku

Služba LISTST vrací v registru A hodnotu 255, je-li logické zařízení pro výstup tisku připraveno přijmout znak, hodnotu 0 v opačném případě. Příslušný podprogram může vracet vždy nulu, neboť standardním systémem není využíván, ale tento způsob realizace nelze doporučit.

Poznámka: U podprogramů pro znakový vstup a výstup se často předpokládá, že příslušná zařízení jsou vždy připravena přijmout, resp. vyslat znak. Pokud tomu tak není (např. zařízení může ohlásit nepřipravenost), je vhodné zabudovat do modulu BIOS příslušné hlášení s možností opravy a opakování operace, namísto prostého návratu s hodnotou 0.

6.1.3 Služby pro obsluhu diskových jednotek

Operace s diskovými jednotkami se provádí postupným voláním podprogramů, které určují číslo disku, číslo stopy, číslo sektoru a adresu DMA, po kterých následuje žádost o čtení, resp. zápis jednoho sektoru. Nastavení čísla stopy a sektoru se provádí pokaždé znovu, určení čísla disku a adresy DMA zůstává platné až do nového nastavení.

Z hlediska modulu BDOS jsou data na diskové jednotce uložena ve větách délky 128 slabik. Jestliže je délka sektoru stejná jako délka věty, odpovídají věty přímo sektorům. Pokud je fyzická délka sektoru větší, musí

modul BIOS obsahovat tzv. *blokovací algoritmus*, který zajišťuje skládání a rozkládání logických vět do, resp. z fyzických sektorů.

SELDSK – Výběr diskové jednotky (SELect DiSK)

K jednotce, jejíž číslo je zadáno v registru C, se budou vztahovat následující operace. Služba SELDSK vrací v registrovém páru HL adresu tabulky DPH (Disk Parameter Header). Jestliže jednotka požadovaného čísla neexistuje, vrací se hodnota 0.

Poznámka: Vybírána je logická disková jednotka. Každá logická jednotka je svázána s fyzickou jednotkou, přičemž více logických jednotek může být přiřazeno k jedné fyzické. Například velkokapacitní disk může být chápán jako více logických jednotek.

Doporučuje se pozdržet fyzický výběr jednotky do okamžiku vyvolání služby READ (WRITE), neboť služba SELDSK je někdy volána bez následného požadavku na čtení (zápis). Výběr neexistující jednotky (je-li službou SELDSK modulu BIOS vrácena nula) signalizuje modul BDOS zprávou:

BDOS Err on n: Select,

kde n: určuje jednotku, která byla vybírána. Po stisku libovolné klávesy na konzoli je proveden teplý start.

Je-li vybrána neexistující jednotka řídicím příkazem monitoru CCP pro změnu aktuální diskové jednotky, dojde k zablokování systému, neboť po teplém startu je tato jednotka opět vybírána. Situaci řeší pouze studený start (reset) systému.

HOME – Nastavení hlavy vybrané diskové jednotky na stopu 0

SETTRK – Nastavení hlavy vybrané diskové jednotky na stopu zadanou v registrovém páru BC (SET TRacK)

SECTRAN – překlad logických čísel sektorů na čísla fyzická (SECtor TRANslation)

Modul BDOS počítá s logickými čísly sektorů (0 až počet sektorů na stopě mínus 1). Pro zlepšení průměrné odezvy systému je zvykem mapovat tato logická čísla na čísla fyzická (1 až počet sektorů na stopě) tak, že logické

sektory nejsou fyzicky uloženy bezprostředně po sobě, ale vždy ob určitý počet sektorů. Na standardních disketách je to např. ob šest sektorů (1, 7, 13, 19, 25, 5 atd.). Toto mapování zajišťuje služba SECTRAN. Logické číslo sektoru v registrovém páru BC slouží jako index do překladové tabulky, jejíž adresa je v registrovém páru DE. Vypočtené číslo fyzického sektoru se vrací v registrovém páru HL. Předpokládá se, že sektorů na stopě není více jak 256 (překladová tabulka má jednoslabikové položky).

SETSEC – Nastavení čísla sektoru na hodnotu uvedenou v registru C resp. registrového páru BC (SET SECtor)

SETDMA – Nastavení adresy DMA na hodnotu zadanou v registrovém páru BC

Adresa DMA určuje adresu kam, resp. odkud se přenášejí data z disku přímým přístupem do operační paměti. Nastavení adresy DMA se provedením služby READ, resp. WRITE nemění.

READ – Čtení věty ze zadané diskové adresy

Služba READ přečte jeden sektor (128 slabik) z vybraného disku, nastavené stopy a sektoru do paměti od nastavené adresy DMA. Po provedení operace se v registru A vrací hodnota 0, bylo-li čtení úspěšné a hodnota 1, byla-li zjištěna neodstranitelná chyba.

WRITE – Zápis věty na zadanou diskovou adresu

Služba WRITE zapíše jeden sektor (128 slabik) z paměti od nastavené adresy DMA na vybraný disk, nastavenou stopu a sektor. Po provedení operace se v registru A vrací hodnota 0, pokud byl zápis úspěšný a hodnota 1, byla-li zjištěna neodstranitelná chyba.

Poznámka: Signalizace chyb diskových jednotek je v systému CP/M poměrně slabá. Chybná operace, signalizovaná nenulovým obsahem registru A, se projeví vypsáním zprávy:

BDOS Err on n: Bad Sector

a uživatel má pouze dvě možnosti. Stiskem klávesy "konec řádky" (CR) chybu ignorovat a stiskem ^C vyvolat teplý start systému. Proto je rozumné

přímo do modulu BIOS zabudovat mechanismus hlášení chyb a reakce na ně. Při zjištění chyby se operace nejprve několikrát (cca 5krát) zopakuje. Při neúspěchu se signalizuje chyba s identifikací jednotky, stopy, sektoru a druhu chyby. Poté lze nechat na uživateli, zda operaci zopakovat nebo ukončit. Reakce na neopravitelnou chybu je závislá na tom, kde se chyba vyskytla.

Disketa porušená v systémové oblasti může být normálně používána ve všech jednotkách kromě jednotky A:. Naproti tomu disketa porušená v oblasti adresáře je prakticky nepoužitelná. Je třeba z ní překopírovat všechny soubory, přičemž některé mohou být ztraceny nebo mohou obsahovat neplatná data.

Je-li disketa porušena v datové oblasti, je vhodné soubor, ve kterém se chyba vyskytla, přejmenovat (např. na CRCERR.xxx) a ponechat na disketě. Ještě lepší řešení je vytvořit program, který z fyzické adresy vadného sektoru určí číslo alokačního bloku, do kterého sektor patří a tento alokační blok přidělí speciálnímu souboru.

Při použití pevného disku by měl být v modulu BIOS zabudován mechanismus náhrady vadných sektorů sektory dobrými.

6.2 Tabulky popisu parametrů diskových jednotek

Parametry diskových jednotek jsou popsány tabulkami v modulu BIOS. To umožňuje využívání libovolných diskových jednotek, příp. i dynamickou změnu charakteristiky jednotky (např. přepínání jednoduchá hustota – dvojitá hustota). Tabulky popisu parametrů diskových jednotek se skládají z *hlaviček* (tabulky DPH – Disk Parameter Header) a *bloků parametrů* (tabulky DPB – Disk Parameter Block). Smyslem rozdělení tabulek je snaha o úsporu místa v paměti, neboť některé části tabulek je možno sdílet.

Tabulky popisu parametrů diskových jednotek jsou přístupné ostatním programům přes volání služby SELDSK modulu BIOS, která vrací adresu tabulky DPH v registrovém páru HL (příp. hodnotu 0, pokud zadaná jednotka neexistuje). Součástí tabulky DPH je i odkaz na použitou tabulku DPB. Protože tabulky popisu parametrů jsou umístěny v paměti RWM, lze je dynamicky modifikovat.

6.2.1 Tabulka DPH (Disk Parameter Header)

Název položky	Délka (slabiky)	Význam
XLT	2	adresa překladové tabulky
–	6	pracovní oblast modulu BDOS
DIRBUF	2	adresa vyrovnávací paměti
DPB	2	adresa tabulky DPB
CSV	2	adresa zóny kontrolního součtu
ALV	2	adresa bitové alokační mapy.

Význam jednotlivých položek tabulky DPH je následující:

XLT adresa tabulky pro překlad logických čísel sektorů na čísla fyzická nebo hodnota 0, jestliže se překlad neprovádí. Tabulka může být uložena kdekoliv v paměti modulu BIOS a její délka je SPT slabik (viz tab. DPB).

DIRBUF adresa vyrovnávací paměti délky 128 slabik pro operace s adresářem. Vyrovnávací paměť může být společná pro všechny jednotky.

DPB adresa tab. DPB. Tab. DPB (Disk Parameter Block) obsahuje charakteristické hodnoty jednotky. Pro jednotky se stejnými charakteristikami může být tab. DPB společná.

CSV adresa pracovní oblasti pro uložení kontrolních součtů vět adresáře při kontrole neregulární výměny média. Pracovní oblast musí být samostatná pro každou jednotku. Její délka je CSK slabik (viz tab. DPB).

ALV adresa pracovní oblasti pro uložení bitové mapy obsazení diskové jednotky. Oblast musí být samostatná pro každou jednotku a její délka je (DSM/8)+1 slabik (viz tab. DPB).

Pro každou logickou jednotku musí být zvláštní tabulka DPH. Pro jednoduchost jsou tabulky uloženy bezprostředně za sebou v pořadí odpovídajícím označení jednotek. Tabulky musí být umístěny v paměti typu RWM, neboť jejich část je využívána modulem BDOS jako pracovní zóna.

6.2.2 Tabulka DPB (Disk Parameter Block)

Název položky	Délka (slabiky)	Význam
SPT	2	počet vět na stopu (Sectors per Track)
BSH	1	dvojkový logaritmus počtu vět v alokačním bloku (Block SHift)
BLM	1	počet vět v alokačním bloku minus 1 (Block Mask)
EXM	1	počet log. rozšíření adresovaných jednou položkou adresáře (EXtent Mask)
DSM	2	celková kapacita diskové jednotky v alokačních blocích mínus 1 (Disk Size Mask)
DRM	2	max.počet položek adresáře (DiRectory Mask)
AL0	1	bitová mapa alokačních bloků
AL1	1	adresáře (ALlocated)
CKS	2	počet sektorů adresáře testovaných na výměnu média (Check Sum)
OFF	2	počet rezervovaných stop (OFFset).

Význam položek tabulky DPB je následující:

SPT počet logických vět na jedné fyzické stopě. Je-li délka sektoru 128 slabik, počet logických vět odpovídá počtu sektorů na stopě. Je-li délka sektoru větší, je počet logických vět roven součinu počtu fyzických sektorů na stopě a počtu vět v jednom sektoru. V případě oboustranné diskety bývá zvykem považovat oba povrchy za jednu stopu s dvojnásobnou kapacitou. Podobně lze řešit i případ více povrchů.

BSH dvojkový logaritmus počtu vět v alokačním bloku.

BLM počet vět v alokačním bloku zmenšený o 1.

EXM počet logických rozšíření (16 KB) adresovaných jednou položkou adresáře zmenšený o 1. Je určen velikostí alokačního bloku (BLS – BLock Size) a celkovým počtem alokačních bloků (DSM) podle tab. 6.2.

Tab. 6.2 Hodnota EXM v závislosti na BLS a DSM

BLS	DSM<256	DSM>255
1 KB	0	–
2 KB	1	0
4 KB	3	1
8 KB	7	3
16 KB	15	7

DRM maximální počet položek adresáře zmenšený o 1. Adresář zabírá vždy celistvý počet alokačních bloků. Každá věta adresáře obsahuje čtyři položky. Proto:

$$DRM = (X*(BLM+1)*4) - 1,$$

kde X je počet alokačních bloků adresáře.

AL0, AL1 první dvě slabiky bitové mapy určující alokační bloky rezervované pro adresář. Jednička v 7. bitu ALO rezervuje 0. blok, v 6. bitu 1. blok atd. až jednička v 0. bitu AL1 rezervuje 15. blok pro adresář.

CKS počet sektorů adresáře testovaných na neregulární výměnu média. U výměnného média má obvykle hodnotu (DRM+1)/4, u pevného média je hodnota CKS rovna 0.

OFF počet stop rezervovaných na počátku diskové jednotky.

Poznámka: Konstanta OFF umožňuje rezervovat na počátku disku místo pro uložení systému. Kromě toho lze konstantu OFF spolu s konstantou DSM využít k rozdělení velkokapacitního disku na více logických jednotek.

Tabulka DPB charakterizuje příslušnou logickou jednotku. Hodnoty v ní obsažené využívá modul BDOS při transformaci logické adresy věty (číslo alokačního bloku, pořadí věty v něm) na její adresu fyzickou (číslo stopy, číslo sektoru). Má-li více logických jednotek stejné parametry, mohou být popsány jednou tabulkou DPB. Tabulky DPB nejsou modulem BDOS modifikovány a mohou proto být uloženy v paměti ROM.

Pro standardní 8palcové diskety formátu IBM 3740 (77 stop / 26 sektorů / 128 slabik) vypadá deklarace tabulek DPH a DPB následovně:

```
BPBASE      EQU     $           ; počátek hlaviček tabulek popisů
                                ; diskových jednotek
; hlavička popisu parametrů jednotky 0
DPH0:       DW      XLT         ; adresa překladové tabulky
            DW      0,0,0       ; pracovní oblast BDOS
            DW      DIRBUF      ; adresa zóny pro adresář
            DW      DPB         ; adresa tabulky DPB
            DW      CSV0        ; adresa zóny kontrolního součtu
            DW      ALV0        ; adresa bitové mapy

; hlavička popisu parametrů jednotky 1
DPH1:

XLT         EQU     $           ; překladová tabulka
            DB      1,7,13,19,25,5,11,17
            DB      23,3,9,15,21,2,8,14
            DB      20,26,6,12,18,24,4,10
            DB      16,22
; tabulka parametrů pro formát IBM 3740
DPB         EQU     $
            DW      26          ; SPT  - počet sektorů na stopě
            DB      3           ; BSH  - dvojkový log. počtu vět
                                ;        v alokačním bloku
            DB      7           ; BLM  - počet vět v alokačním
                                ;        bloku - 1
            DB      0           ; EXM  - počet log. rozšíření
                                ;        adresovaných jednou
                                ;        položkou adresáře
            DW      242         ; DSM  - celkový počet alokačních
                                ;        bloků na disku - 1
            DW      63          ; DRM  - maximální počet položek
                                ;        adresáře - 1
            DB      0C0H        ; ALO  - bitová mapa alokačních
            DB      0           ; AL1    bloků adresáře
            DW      16          ; CKS  - počet sektorů adresáře
                                ;        kontrolovaných při zápisu
                                ;        (zde celý adresář)
```

```
            DW      2       ; OFF - počet rezervovaných stop
                            ;       vyrovnávací paměti
DIRBUF:     DS      128     ; zóna pro větu adresáře
ALVO:       DS      31      ; zóna pro bitovou mapu diskové
                            ;   jednotky (zde maximálně
                            ;   31*8=248 alokačních bloků)
CSVO:       DS      16      ; zóna pro kontrolní součty
                            ;   každé věty adresáře
```

6.3 Blokovací algoritmus

Jestliže je délka fyzického sektoru odlišná od délky logické věty (128 slabik), musí modul BIOS zahrnovat *blokovací algoritmus*. Blokovací algoritmus zajišťuje transformaci mezi větami a sektory prostřednictvím vyrovnávací paměti s kapacitou na jeden nebo více sektorů.

Na způsobu řešení blokovacího algoritmu silně závisí efektivnost práce systému. Aby bylo možno realizovat blokovací algoritmus efektivně, předává modul BDOS při vyvolání služby WRITE v registru C jako parametr jednu z následujících hodnot.

0 – normální zápis

1 – zápis do adresáře

2 – první zápis do nealokovaného bloku.

Hodnota 0 indikuje, že je zapisována věta do již zapsaného alokačního bloku. To znamená, že sektor, který obsahuje zapisovanou větu, je třeba nejprve přečíst z disku do vyrovnávací paměti a v ní pak přepsat úsek odpovídající zapisované větě. Nedodržení této zásady by při použití přímého přístupu do souboru způsobilo ztrátu již zapsaných dat.

Zápis obsahu vyrovnávací paměti na disk je vhodné odložit, neboť může následovat zápis věty do stejného sektoru. Zápis věty do adresáře končí zpracování výstupního souboru. Musí být proto proveden okamžitě. Pokud je použit algoritmus s více vyrovnávacími pamětmi, je třeba rovněž provést všechny odložené zápisy.

Zápis do nealokovaného bloku je signalizován při zápisu první věty do doposud nepřiděleného alokčního bloku. Termín "první věta" je třeba chápat v časovém smyslu. Při použití přímého přístupu se může jednat o libovolnou

větu bloku. Rovněž je třeba si uvědomit, že všechny následující zápisy do tohoto bloku jsou již označeny jako normální.

Možné řešení blokovacího algoritmu s jednou vyrovnávací pamětí je uvedeno v následujícím programu. Toto řešení vychází z předpokladu, že nejčastější případ je sekvenční zpracovávání jednoho souboru. Pro tento případ je efektivní. Při zápisu do nealokovaného bloku se předpokládá, že zápis směřuje do první věty nealokovaného bloku. V tomto okamžiku je zaznamenáno, kolik ještě nezapsaných vět obsahuje tento čerstvě přidělený alokační blok a jakou diskovou adresu má následující věta bloku.

Při normálním zápisu se nejprve testuje, nejedná-li se o pokračování v zápise nealokovaného bloku (tj. zbývají-li nějaké věty a souhlasí-li disková adresa). Jestliže ano, zpracuje se normální zápis obdobně jako zápis do nealokovaného bloku. Jestliže ne, je třeba provést předečtení, (pokud vyrovnávací paměť požadovaný sektor neobsahuje). Zápis do adresáře se provádí vždy jako zápis do alokovaného bloku. Přitom se zápis neodkládá, ale okamžitě provádí. Efektivita uvedeného algoritmu rapidně klesá, je-li zpracováváno více souborů současně.

```
; ***************************************************************
; *                                                             *
; *       Blokovaci algoritmus pro CP/M verze 2.x               *
; *       s vyrovnavaci pameti na jeden blok                    *
; *                                                             *
; ***************************************************************
;
smask   macro hblk
;;      vypocet dvojkoveho logaritmu hblk, vysledek v @x
;;      (tj. po navratu 2 ** @x = hblk)
@y      SET   hblk
@x      SET   0
;;      while @y<>1 do begin @x:=@x+1; @y:=@y div 2 end
        rept  8
        IF    @y eq 1
        exitm
        ENDIF
```

```
@y      SET   @y shr 1
@x      SET   @x + 1
        endm
        endm
;*****************************************************************
;*                                                               *
;*      konstanty logickeho a fyzickeho disku                    *
;*                                                               *
;*****************************************************************
blksiz  EQU 2048            ; delka alokacniho bloku
hstsiz  EQU 512             ; delka fyzickeho sektoru
hstspt  EQU 20              ; fyz. sektoru na stopu
hstblk  EQU hstsiz/128      ; log. vet na fyzicky sektor
cpmspt  EQU hstblk * hstspt ; log. vet na stopu
secmsk  EQU hstblk-1        ; maska cisla vety
        smask hstblk
secshf  EQU @x              ; log2 (hstblk)
;
;*****************************************************************
;*                                                               *
;*      priznaky od sluzby write                                 *
;*                                                               *
;*****************************************************************
wrall   EQU 0               ; zapis do alokovaneho bloku
wrdir   EQU 1               ; zapis do adresare
wrual   EQU 2               ; zapis do nealokovaneho bloku
;
;*****************************************************************
;*                                                               *
;*      V nasledujicim textu programu jsou uvedeny pouze         *
;*      ty casti modulu BIOS, ktere je nutno upravit             *
;*      v souvislosti s blokovacim algoritmem.                   *
;*                                                               *
;*****************************************************************
;       tabulky DPH a DPB patri sem
dpbase  EQU $               ; baze tabulek DPH
```

```
boot:
wboot:
        ; inicializace systemu
        XOR  A                  ; 0 do stradace
        LD   (hstact), A        ; vyrovnavaci pamet neaktivni
        LD   (unacnt), A        ; nulovani pocitadla
        RET                     ; nealokovanych zapisu
;
        ; nastaveni stopy 0 vybraneho disku
home:
        LD   A,(hstwrt)         ; zbyva neco k zapisu?
        OR   A
        JP   NZ,homed
        LD   (hstact), A        ; ne, fyzicky disk je pasivovan
homed:
        RET
;
        ; vyber disku
seldsk:
        LD   A, C               ; cislo vybiraneho disku
        LD   (sekdsk), A        ; do sekdsk
        LD   L,A                ; cislo disku do HL
        LD   H,0
        rept 4                  ; vynasobit 16
        ADD  HL,HL
        endm
        LD   DE, dpbase         ; baze tabulek DPB do DE
        ADD  HL,DE              ; adresa tabulky DPB do HL
        RET
        ; nastaveni cisla stopy na hodnotu zadanou v BC
settrk:
        LD   H,B
        LD   L,C
        LD   (sektrk),HL        ; cislo stopy do sektrk
        RET
```

```
        ; nastaveni cisla sektoru
        ; cislo sektoru v registru C
setsec:
        LD    A,C
        LD    (seksec),A          ; cislo sektoru do seksec
        RET
;
        ; nastaveni adresy DMA
        ; hodnota DMA v registrovem paru BC
setdma:
        LD    H,B
        LD    L,C
        LD    (dmaadr),Hl         ; adresa DMA do dmaadr
        RET
;
        ; preklad logickych cisel sektoru na fyzicka
        ; logicke cislo sektoru v BC
sectran:
        LD    H,B                 ; preklad se neprovadi
        LD    L,C                 ; HL=fyzicke=logicke cislo
        RET
;
;*****************************************************************
;*                                                               *
;*      READ: cteni jedne vety                                   *
;*                                                               *
;*****************************************************************
read:
        XOR   A                   ; nuluj citac
        LD    (unacnt),A          ; nealokovanych zapisu
        LD    A,1
        LD    (readop),A          ; priznak operace cteni
        LD    (rsflag),A          ; priznak, ze je nutno cist
        LD    A,wrual
        LD    (wrtype),A          ; povazuje za nealokovany
        JP    rwoper              ; proved operaci
```

```
;*****************************************************************
;*                                                               *
;*        WRITE: zapis jedne vety                                *
;*                                                               *
;*****************************************************************
write:
        XOR  A                  ; 0 do stradace
        LD   (readop),A         ; neni cteci operace
        LD   A,C                ; typ zapisu v C
        LD   (wrtype),A
        CP   wrual              ; zapis do nealokovaneho bl.
        JP   NZ,chkuna          ; ne, test na pokracovani
                                ; v nealokovanem zapise
;       jinak zapis do nealokovaneho bloku a nastaveni parametru
;
; Poznamka:
;       Tento algoritmus predpoklada, ze prvni zapis do nealokovaneho
; bloku smeruje do 1. vety tohoto bloku. Algoritmus nezapracuje spravne,
; nastane-li nasledujici (pomerne vykonstruovana) situace:
;       Primym pristupem byl vytvoren soubor, ktery neni celistvy - za
; nealokovanym blokem je alokovany blok s platnymi daty.
;       Primym pristupem je do nealokovaneho bloku zapsana veta jina nez
; prvni.
;       Primym nebo sekvencnim pristupem se postupne zapisuji dalsi vety.
;       Pocet prepsanych vet v alokovanem bloku neni nasobkem poctu vet
; v jednom fyzickem sektoru.
;
;       V tomto pripade nastane chyba, nebot zapis do alokovaneho bloku
; bude interpretovan jako zapis do nealokovaneho.
;
        LD   A,blksiz/128       ; pocet nealokovanych zapisu
        LD   (unacnt),A         ; !! chyba, neni-li 1. veta
        LD   A,(sekdsk)         ; vybrany disk
        LD   (unadsk),A         ; unadsk = sekdsk
        LD   HL,(sektrk)
```

```
        LD    (unatrk),HL       ; unatrk = sectrk
        LD    A,(seksec)
        LD    (unasec),A        ; unasec = seksec
;
chkuna:
        ; test na pokracovani v nealokovanem zapisu
        LD    A,(unacnt)        ; citac nealokovanych
        OR    A                 ; zapisu
        JP    Z,alloc           ; je nulovy
;
; ne, muze se jednat o pokracovani v nealokovanem zapise
        DEC   A                 ; unacnt = unacnt-1
        LD    (unacnt),A
        LD    A,(sekdsk)        ; stejny disk?
        LD    HL,unadsk
        CP    (HL)              ; sekdsk = unadsk?
        JP    NZ,alloc          ; ne
;
;       disky jsou stejne - stejne stopy?
        LD    HL,(unatrk)
        CALL  stcpm             ; sektrk = unatrk?
        JP    NZ,alloc          ; ne
;
;       stopy jsou stejne - stejne sektory?
        LD    A,(seksec)
        LD    HL,unasec
        CP    (HL)              ; seksec = unasec?
        JP    NZ,alloc          ; ne
;
;       shoda, priprav adresu vety pro pristi zapis
        INC   (HL)              ; unasec = unasec+1
        LD    A,M               ; konec stopy?
        CP    CPMSPT
        JP    C,noovf           ; ne
;
```

```
;       dalsi stopa
            LD    (HL),0              ; unasec = 0
            LD    HL,(unatrk)
            INC   H
            LD    (unatrk),HL         ; unatrk = unatrk+1
;
noovf:
        ; doslo ke shode, neni treba predecitat
            XOR   A                   ; 0 do stradace
            LD    (rsflag),A          ; rsflag = 0
            JP    rwoper
;
alloc:
        ; zapis do alokovaneho, nutno predecist
            XOR   A
            LD    (unacnt),A          ; unacnt = 0
            INC   A
            LD    (rsflag),A          ; rsflag = 1
;
;
;*****************************************************************
;*                                                               *
;*       Spolecny kod pro READ a WRITE                           *
;*                                                               *
;*****************************************************************
rwoper:
            XOR   A
            LD    (erflag),A          ; nuluj priznak chyby
            LD    A,(seksec)          ; vypocet cisla fyz. sektoru
            rept  secshf
            OR    A
            RRA
            endm
            LD    (sekhst),A          ; cislo fyzickeho sektoru
;
```

```
;       vyrovnavaci pamet aktivni?
            LD    HL,hstact
            LD    A,(HL)
            LD    (HL),1              ; vzdy se stane aktivni
            OR    A                   ; byla aktivni?
            JP    Z, filhst           ; precti sektor, jestlize ne
;
;       vyrovnavaci pamet je aktivni
;       obsahuje pozadovany sektor?
            LD    A,(sekdsk)
            LD    HL,hstdsk           ; stejny disk?
            CP    (HL)                ; sekdsk = hstdsk
            JP    NZ,nomatch
;
;       stejny disk - stejna stopa?
            LD    HL,hsttrk
            CALL stcmp                ; sektrk = hsttrk?
            JP    NZ,nomatch
;
;       stejny disk, stejna stopa - stejny sektor?
            LD    A,(sekhst)
            LD    HL,hstsec           ; sekhst - hstsec?
            CP    (HL)
            JP    Z,match             ; ano
;
nomatch
        ; pozadovan jiny sektor nez ten ve vyrovnavaci pameti
            LD    A,(hstwrt)          ; je treba sektor z VP zapsat?
            OR    A
            CALL NZ,writehst          ; jestlize ano, zapis ho
;
filhst:
        ; napln vyrovnavaci pamet
            LD    A,(sekdsk)
            LD    (hstdsk),A
            LD    HL,(sektrk)
```

```
            LD    (hsttrk),HL
            LD    A,(sekhst)
            LD    (hstsec),A
            LD    A,(rsflag)          ; je treba cist
            OR    A
            CALL NZ,readhst           ; jestlize ano, precti sektor
            XOR A
            LD    (hstwrt),A          ; nuluj priznak odlozeneho zapisu
;
match:
        ; presun data do nebo z vyrovnavaci pameti
            LD    A,(seksec)          ; nizsi bity cisla vety
            AND   secmsk
            LD    L,A                 ; po vynasobeni 128
            LD    H,0
            rept  7
            ADD   HL,HL
            endm
; v registru hl je relativni adresa ve vyrovnavaci pameti
            LD    DE,hstbuf           ; urcuji posunuti
            ADD   HL,DE               ; HL = adresa vety v hstbuf
            EX    DE,HL               ; do DE
            LD    HL,(dmaadr)         ; adresa DMA
            LD    C,128               ; delka presunu
            LD    A,(readop)          ; kterym smerem?
            OR    A
            JP    NZ,rwmove           ; preskok, je-li cteni
;
;       zapis, oznac a obrat smer
            LD    A,1
            LD    (hstwrt),A          ; hstwrt = 1
            EX    DE,HL
;
rwmove:
        ; presun - C delka (128), DE vychozi, HL cilova adr.
            LD    A,(DE)
```

```
        INC   DE
        LD    (HL),A
        INC   HL
        DEC   C                     ; cykl 128krat
        JP    NZ, rwmove
;
;       data byla presunuta z/do hstbuf
        LD    A,(wrtype)            ; typ zapisu
        CP    wrdir                 ; do adresare?
        LD    A,(erflag)            ; vysledek operace
        RET   NZ                    ; jestlize ne, konec
;       zapis do adresare treba ihned provest
        OR    A                     ; byly chyby?
        RET   NZ                    ; jestlize ano, konec
        XOR   A
        LD    (hstwrt),A            ; zrus priznak odlozeneho zapisu
        CALL writehst               ; a zapis proved
        LD    A,(erflag)
        RET
;
;*****************************************************************
;*                                                               *
;*       Podprogram na 16ti bitove porovnani                     *
;*                                                               *
;*****************************************************************
stcmp:
        ; HL = ^unatrk nebo ^hsttrk, porovnej s sektrk
        EX    DE,HL
        LD    HL,sektrk
        LD    A,(DE)
        CP    (HL)                  ; nizsi byty stejne?
        RET   NZ                    ; navrat, jestlize ne
;       nizsi byty stejne, testuj vyssi
        INC   DE
        INC   HL
        LD    A,(DE)
```

```
        CP    (HL)                ; nastav priznaky
        RET
;
;*****************************************************************
;*                                                               *
;*      WRITEHST provadi fyzicky zapis sektoru                   *
;*      READHST provadi fyzicke cteni                            *
;*                                                               *
;*****************************************************************
writehst:
        ; hstdsk = cislo disku, hsttrk = cislo stopy
        ; hstsec = cislo sektoru. zapis "hstsiz" bytu z hstbuf a doslo-li
        ; k chybe, nastav erflag na kod chyby<>0
        RET
;
readhst:
        ; hstdsk = cislo disku, hsttrk = cislo stopy
        ; hstsec = cislo sektoru. cti "hstsiz: bytu do hstbuf a doslo-li
        ; k chybe nastav erflag na kod chyby<>0
        RET
;
;*****************************************************************
;*                                                               *
;*      Neinicializovana oblast pameti RWM                       *
;*                                                               *
;*****************************************************************
; zadana diskova adresa vety
;
sekdsk: DS    1                 ; disk
sektrk: DS    2                 ; stopa
seksec: DS    1                 ; sektor
;
; diskova adresa fyzickeho sektoru
;
hstdsk: DS    1                 ; disk
```

```
hsttrk:  DS   2                ; stopa
hstsec:  DS   1                ; sektor
;
sekhst:  DS   1                ; cislo pozadovaneho fyzickeho
                               ; sektoru
hstact:  DS   1                ; priznak hstbuf aktivni
hstwrt:  DS   1                ; priznak odlozeneho zapisu
;
unacnt:  DS   1                ; citac nealokovanych zapisu
; diskova adresa dalsi vety v nealokovanem bloku
unadsk:  DS   1                ; disk
natrk:   DS   2                ; stopa
unasec:  DS   1                ; sektor
;
erflag:  DS   1                ; kod chyby
rsflag:  DS   1                ; priznak pozadavku na cteni
readop:  DS   1                ; 1 jestlize cteci operace
wrtypew: DS   1                ; typ zapisove operace
dmaadr:  DS   2                ; adresa DMA
hstbuf:  DS   hstsiz           ; vyrovnavaci pamet
;
;*****************************************************************
;*                                                               *
;*       ENDEF  makro nebo definice pracovnich bunek             *
;*              patri sem                                        *
;*                                                               *
;*****************************************************************
         END
```

6.4 Generování systému CP/M

Generováním systému CP/M se rozumí přizpůsobení operačního systému CP/M konkrétním technickým prostředkům daného počítače. Přizpůsobení se provádí tak, že se moduly CCP a BDOS rekonfigurují pro požadovanou velikost operační paměti a doplní se k nim modul BIOS a systémový zavaděč.

Při generování systému se vychází z tzv. *distribuční diskety systému*, která obsahuje v systémové oblasti (stopy 0 a 1) *distribuční obraz* systému a v datové oblasti *služební programy* SYSGEN, MOVCPM a sadu prostředků pro přípavu a ladění programů v asembleru příslušného procesoru, např. ASM, LOAD a DDT (pro 8080), nebo M80, L80 a ZSID (pro Z80).

Pokud generujeme systém pomocí křížového programového vybavení na jiném počítači, je nutno mít k dispozici mimo distribuční disketu, křížové prostředky pro vytváření programů v kódu cílového procesoru a prostředky pro čtení a záznam na disk ve formátu odpovídajícím struktuře diskového média CP/M.

Generování systému probíhá ve dvou etapách:
- generování úrovně 1 (vytvoření pracovní verze systému ve stavu, kdy nejsou k dispozici služby systému),
- generování úrovně 2 (vytvoření konečné verze systému ve stavu, kdy je k dispozici pracovní verze systému vytvořená v etapě 1).

Pro generování systému je nutno podrobně znát technické prostředky cílového počítače (činnost po zapnutí, ovládání konzole, diskových jednotek atd.)

Generování úrovně 1

Pro vytvoření pracovní verze systému je nutno mít možnost vytvářet programy v kódu cílového procesoru (např. překládat programy z asembleru a přeložené programy zavádět do počítače), pro který systém generujeme. Pro zavádění programů je třeba mít k dispozici technický zavaděč. Rovněž musíme rozhodnout, jakým způsobem se bude do paměti zavádět modul BIOS. Generování úrovně 1 se skládá z následujících kroků:

1. Vytvoříme program GETSYS, který *čte systémovou oblast diskety* (stopy 0 a 1 – str. 236) do operační paměti (od adresy 3380H pro standardní distribuční verzi systému, která je konfigurována pro paměť o rozsahu 20 KB). Program GETSYS nechť začíná na adrese 100H. Kostra programu GETSYS je uvedena dále.

2. Ověříme činnost programu GETSYS na pracovní disketě a zkontrolujeme, zda se obsah diskety programem GETSYS neporušil.

3. Obdobně vytvoříme program PUTSYS, který *zapisuje použitou oblast paměti* (od adresy 3380H), tj. obraz systému na disketu. Jeho funkci opět ověříme na prázdné pracovní disketě. Kostra programu PUTSYS je opět uvedena dále.

4. Upravíme kostru modulu BIOS uvedenou v dalším textu pro dané konkrétní technické prostředky a dle umístění operačního systému v paměti, kterou máme k dispozici. Realizujeme pouze nejjednodušší verzi diskových operací pro systémovou jednotku (A:) a obsluhu systémové konzole. Také modul BIOS patřičně otestujeme.

5. Pomocí programu GETSYS přečteme operační systém z distribuční diskety do paměti, doplníme modul BIOS a programem PUTSYS uložíme na disketu (pro jistotu na jinou disketu než distribuční).

6. Předáme řízení na počátek modulu BIOS (studený start). Je-li vše v pořádku, systém přečte adresář ze systémové jednotky (A:) a ohlásí se standardní výzvou

```
A>
```

7. Otestujeme pracovní verzi systému. Pokud jsme použili prázdnou disketu (musí být nová, předznačená nebo alespoň v oblasti adresáře popsaná hodnotami 0E5H), může test vypadat např. následovně (vstup zadávaný operátorem je pro názornost uveden malými písmeny):

```
A>dir
NO FILE
A>save 1 x.com
A>dir
A:X COM
A>era x.com
A>dir
NO FILE
A>
```

8. Podle zvoleného způsobu zavádění modulu BIOS do paměti realizujeme základní systémový zavaděč (nazývaný také studený zavaděč). Je-li kód modulu BIOS kopírován z paměti typu ROM, doplníme do modulu BIOS

příslušný přesun. V tomto případě je možno vynechat program GETSYS a realizovat přesun přímo v rámci služby BOOT modulu BIOS.

Kostru zavaděče systému příslušně doplníme a umístíme standardním způsobem na disk (viz. str. 236). Tento zavaděč je třeba přeložit od adresy, kam je zaváděn technickým zavaděčem (obvykle umístěným v paměti typu ROM), a poté jej umístit na prvý sektor nulté stopy disku (boot sektor). V obou případech je vhodné dodržet zásadu, že studený start nepřepisuje zónu TPA. Dodržení této zásady oceníme v případech, kdy se program v zóně TPA "zadře". Po studeném startu lze obsah zóny TPA uložit na disk příkazem SAVE a poté prohlédnout pomocí programů DUMP, resp. DDT.

9. Je-li vše v pořádku, uložíme pracovní verzi operačního systému na disketu se soubory (na kopii distribuční diskety). Tím máme všechny programy pro generování úrovně 2 k dispozici na této disketě.

```
;
;          KOSTRA MODULU CBIOS PRO CP/M VERZE 2 . x
;
MSIZE   EQU  20                 ; VELIKOST PAMETI PRO DANOU
                                ; KONFIGURACI V KILOBYTECH
;
;       "BIAS" JE POSUN ADRES VZHLEDEM K ADRESE 3400H
;       PRO INSTALACI SYSTEMU CP/M NA POCITACI
;       S PAMETOVOU KONFIGURACI VETSI NEZ 20 KB
;
BIAS    EQU     (MSIZE-20)*1024
CCP     EQU     3400H+BIAS; ZACATEK MODULU CCP
BDOS    EQU     CCP+806H  ; ZACATEK MODULU BDOS
BIOS    EQU     CCP+1600H ; ZACATEK MODULU BIOS
CDISK   EQU     0004H     ; CISLO AKTUALNIHO DISKU
IOBYTE  EQU     0003H     ; INTEL IOBYTE
;
        ORG     BIOS      ; POCATECTNI ADRESA
NSECTS  EQU     ($-CCP)   ; POCET SEKTORU PRO
                          ; TEPLY START
;
;
```

```
;       SKOKOVY VEKTOR NA JEDNOTLIVE PODPROGRAMY
                JP BOOT         ; STUDENY START
WBOOTE          JP WBOOT        ; TEPLY START
                JP CONST        ; STAV KONZOLE
                JP CONIN        ; VSTUP ZNAKU Z KONZOLE
                JP CONOUT       ; VYSTUP ZNAKU NA KONZOLI
                JP LIST         ; VYSTUP ZNAKU NA TISKARNU
                JP PUNCH        ; VYSTUP ZNAKU NA DEROVAC
                JP READER       ; VSTUP ZNAKU ZE SNIMACE
                JP HOME         ; PRESUN HLAVY DO
                                ; VYCHOZI POZICE
                JP SELDSK       ; VYBER DISKU
                JP SETTRK       ; NASTAVENI CISLA STOPY
                JP SETSEC       ; NASTAVENI CISLA SEKTORU
                JP SETDMA       ; NASTAVENI ADRESY DMA
                JP READ         ; CTENI Z DISKU
                JP WRITE        ; ZAPIS NA DISK
                JP LISTST       ; STAV TISKARNY
                JP SECTRAN      ; PREKLAD CISLA SEKTORU
;
;       PEVNE DATOVE TABULKY PRO STANDARDNI DISKOVY
;       SYSTEM SE CTYRMI 8" PRUZNYMI DISKY
;
;       TABULKA DPH PRO DISK 0
DPBASE:DW       TRANS,0000H
          DW    0000H,0000H
          DW    DIRBF, DPBLK
          DW    CHK00,ALL00
;       TABULKA DPH PRO DISK 1
          DW    TRANS,0000H
          DW    0000H,0000H
          DW    DIRBF, DPBLK
          DW    CHO01,ALL01
;       TABULKA DPH PRO DISK 2
          DW    TRANS,0000H
          DW    0000H,0000H
```

```
            DW      DIRBF, DPBLK
            DW      CHK02,ALL02
;       TABULKA DPH PRO DISK 3
            DW      TRANS,0000H
            DW      0000H,0000H
            DW      DIRBF,DPBLK
            DW      CHK03,ALL03
;
;       TABULKA PRO PREKLAD CISEL SEKTORU
TRANS:  DB      1,7,13,19       ; SEKTORY 1,2,3,4
            DB      25,5,11,17      ; SEKTORY 5,6,7,8
            DB      23,3,9,15       ; SEKTORY 9,10,11,12
            DB      21,2,8,14       ; SEKTORY 13,14,15,16
            DB      20,26,6,12      ; SEKTORY 17,18,19,20
            DB      18,24,4,10      ; SEKTORY 21,22,23,24
            DB      16,22           ; SEKTORY 25,26
;
DPBLK:  ; TABULKA DPB SPOLECNA PRO VSECHNY DISKY
            DW      26              ; SPT
            DB      3               ; BSH
            DB      7               ; BLM
            DB      0               ; EXM
            DB      242             ; DSM
            DB      63              ; DRM
            DB      192             ; AL0
            DB      0               ; AL1
            DW      16              ; CHS
            DW      2               ; OFF
;
;       KONEC PEVNYCH TABULEK
;

;       PODPROGRAMY REALIZUJICI JEDNOTLIVE SLUZBY

BOOT:   ; V NEJJEDNODUSSIM PRIPADE POUZE INICIALIZACE
            ; PARAMETRU
```

```
        XOR     A               ;
        LD      (IOBYTE),A      ; NULOVANI IOBYTE
        LD      (CDISK),A       ; NASTAVENI DISKU 0
                                ; A UZIVATELE 0
        JP      GOCPM           ; INICIALIZACE A VOLANI CP/M
;
WBOOT:
        LD      SP,80H          ; NASTAVENI SP
        LD      C,0             ; VYBER DISKU 0 (SYSTEMOVE
        CALL    SELDSK          ; DISKOVE JEDNOTKY)
        CALL    HOME            ; PRESUN HLAVY NA STOPU 0
;
        LD      B,NSECTS        ; B=POCET SEKTORU PRO
                                ; TEPLY START
        LD      C,0             ; C=CISLO STOPY
        LD      D,2             ; D=CISLO SEKTORU KTERY
                                ; SE MA CIST
;
;       CTENI ZACINA NA STOPE 0 SEKTOR 2,
;       SEKTOR 1 (BOOT SEKTOR) OBSAHUJE SYSTEMOVY
;       ZAVADEC
;
        LD      HL,CCP          ; POCATECNI ADRESA CP/M
LOADL: ; CTENI JEDNOHO SEKTORU
        PUSH    BC              ; ULOZENI CITACE SEKTORU
                                ; A STOPY
        PUSH    DE              ; ULOZENI CISLA SEKTORU,
                                ; KTERY SE MA CIST
        PUSH    HL              ; ULOZENI ADRESY DMA
        LD      C,D             ; CISLO SEKTORU DO C
        CALL    SETSEC          ; NASTAVENI CISLA SEKTORU
        POP     BC              ; ADRESA DMA ZPET DO B
        PUSH    BC              ; ZPET NA ZASOBNIK
        CALL    SETDMA          ; NASTAVENI ADRESY DMA Z B
;
```

```
;       NASTAVEN DISK, STOPA, SEKTOR, DMA
        CALL    READ
        CP      00H             ; NASTALA CHYBA?
        JNZ     WBOOT           ; POKUD ANO, ZOPAKUJ
                                ; ZAVLECENI
;
;       DALSI SEKTOR:
        POP     HL
        LD      DE,128          ; DMA:=DMA+128
        ADD     HL,DE           ; NOVA ADRESA DMA JE V HL
        POP     DE              ; VYZVEDNUTI CISLA SEKTORU
        POP     BC              ; VYZVEDNUTI POCTU SEKTORU,
                                ; KTERE JE JESTE NUTNO PRECIST
        DEC     B               ; ZMENSENI O 1
        JP      Z,GOCPM         ; PRECHOD DO SYSTEMU, BYL-LI
                                ; PRENESEN CELY OBRAZ
                                ; SYSTEMU
;
;       JE POTREBA CIST DALSI SEKTOR
        INC     D
        LD      A,D             ; SEKTOR=27?
        CP      27
        JP      C,LOADL         ; CARRY SE GENERUJE, KDYZ
                                ; SEKTOR JE MENSI NEZ 27
;
;       KONEC STOPY PRECHOD NA DALSI
        LD      D,1             ; PRVNI SEKTOR DALSI STOPY
        INC     C               ; STOPA:=STOPA+1
;
;       ULOZENI REGISTRU A ZMENA STOP
        PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    SETTRK          ; NASTAVENI STOPY Z REG. C
        POP     HL
        POP     DE
```

```
        POP     BC
        JP      LOADL       ; CTI DALSI SEKTOR
;       KONEC ZAVLEKANI, NASTAV PARAMETRY A VSTUP DO
;       SYSTEMU
GOCPM:
        LD      A,0C3H      ; C3 JE INSTRUKCE JP
        LD      (0),A       ; JP PRO TEPLY START
        LD      HL,WBOOTE   ; VSTUP PRO TEPLY START
        LD      (1),HL      ; ULOZENI ADRESY PRO
                            ; SKOK NA ADRESE 0
        LD      (5),A       ; JP PRO SKOK DO MODULU
                            ; BDOS
        LD      HL,BDOS     ; VSTUPNI ADRESA MODULU
                            ; BDOS
        LD      (6),HL      ; ULOZENI ADRESY PRO
                            ; SKOK NA ADRESE 5
        LD      HL,80H      ; STANDARTNI ADRESA DMA
        CALL    SETDMA
;
        EI                  ; POVOLENI PRERUSENI
        LD      A,(CDISK)   ; CISLO AKTUALNIHO
                            ; DISKU A UZIVATELE
        LD      C,A         ; JE PREDANO MODULU CCP
        JP      CCP         ; VSTUP DO SYSTEMU
;
;
;       PODPROGRAMY PRO V/V
;       MUSI BYT DOPLNENY UZIVATELEM
;
CONST:  ; STAV SYSTEMOVE KONSOLE:
        ; VRACI HODNOTU 0FFH, JE-LI ZNAK PRIPRAVEN,
        ; JINAK VRACI HODNOTU 0
        DS      10H         ; MISTO PRO STAV. PODPROGRAM
        LD      A,0
        RET
;
```

```
CONIN:  ; CTENI ZNAKU ZE SYSTEMOVE KONZOLE DO
        ; REGISTRU A
        DS      10H         ; MISTO PRO VSTUPNI
                            ; PODPROGRAM
        AND     7FH         ; NULOVANI PARITNIHO BITU
        RET
;
CONOUT: ; VYSTUP ZNAKU Z REGISTRU C
        ; NA SYSTEMOVOU KONZOLI
        LD      A,C         ; PRESUN DO AKUMULATORU
        DS      10H         ; MISTO PRO VYSTUPNI
                            ; PODPROGRAM
        RET
;
LIST: ; VYSTUP ZNAKU Z REGISTRU C NA ZARIZENI PRO TISK
        LD      A,C         ; ZNAK DO A
        RET                 ; PRAZDNY PODPROGRAM
;
LISTST: ; VRAT STAV TISKOVEHO ZARIZENI
        XOR     A           ; VZDY PRIPRAVEN
        RET
;
PUNCH: ; DEROVANI ZNAKU Z REGISTRU C
        LD      A,C
        RET                 ; PRAZDNY PODPROGRAM
;
READER: ; CTENI ZNAKU ZE SNIMACE DO REGISTRU A
        LD      A,1AH       ; PROZATIM JEN VSTUP
                            ; ZNAKU EOF
        AND     7FH         ; NULOVANI PARITY
        RET
;
;
;       PODPROGRAMY PRO OBSLUHU DISKU
HOME:   ; PRESUN NA STOPU 00 AKTUALNIHO DISKU
        ; PREVEDENO NA VOLANI SETTRK S PARAMETREM 00
```

```
        LD      C,0
        CALL    SETTRK
        RET
;
SELDSK:; VYBER DISKOVE JEDNOTKY SPECIFIKOVANE
        ; V REGISTRU C
        LD      HL,0000H    ; KOD NAVRATU PRI CHYBE
        LD      A,C
        LD      (DISKNO),A
        CP      4           ; MUSI BYT OD 0 DO 3
        RET     NC          ; JE-LI 4,5, . .
        ; CISLO DISKOVE JEDNOTKY JE VE SPRAVNEM
        ; ROZSAHU
        DS      10          ; MISTO PRO VYBER DISKU
        ; VYPOCET ADRESY ODPOVIDAJICI TABULKY DPH
        LD      A,(DISKNO)
        LD      L,A         ; L:=CISLO DISKU (0,1,2,3)
        LD      H,0         ; NULOVANI H
        ADD     HL,HL       ; *2
        ADD     HL,HL       ; *4
        ADD     HL,HL       ; *8
        ADD     HL,HL       ; *16
        LD      DE,DPBASE
        ADD     HL,DE       ; HL:= DBASE+DISKNO*16
        RET
;
SETTRK:; NASTAV STOPU ZADANOU V REGISTRU C
        LD      A,C
        LD      (TRACK),A
        DS      10H         ; MISTO PRO VYBER STOPY
        RET
;
SETSEC:; NASTAV SEKTOR ZADANY V REGISTRU C
        LD      A,C
        LD      (SECTOR),A
```

```
        DS      10H         ; MISTO PRO VYBER SEKTORU
        RET
;
SECTRAN:
        ; PREKLAD CISLA SEKTORU ZADANEHO V REGISTRO-
        ; VEM PARU BC POMOCI PREKLADOVE TABULKY NA
        ; ADRESE V PARU DE
        EX      DE,HL
        ADD     HL,BC
        LD      L,M
        LD      H,0
        RET
;
SETDMA:; NASTAV ADRESU PRO DMA ZADANOU V REGIS-
        ; TROVEM PARU BC
        LD      L,C             ; NIZSI BYTE ADRESY
        LD      H,B             ; VYSSI BYTE ADRESY
        LD      (DMAAD),HL      ; ULOZENI ADRESY
        DS      10H             ; MISTO PRO VLASTNI
                                ; PROGRAM
        RET
;
READ:   ; PROVEDENI OPERACE CTENI (OBVYKLE PODOBNE
        ; PRO OPERACI ZAPISU, PROTO JE VYHRAZENA PA-
        ; MET POUZE PRO NASTAVENI ZADOSTI READ/WRITE,
        ; NASLEDUJE SPOLECNA CAST PRO CTENI I ZAPIS)
        DS      10H             ; NASTAVENI ZADOSTI READ
        JP      WAITIO          ; PROVEDENI
;
WRITE:  ; PROVEDENI OPERACE ZAPISU
        DS      10H             ; NASTAVENI ZADOSTI WRITE
;
WAITIO:; ZDE JE VSTUP Z READ A WRITE K FYZICKEMU
        ; PROVEDENI OPERACE. POKUD OPERACE PROBEHNE
        ; SPRAVNE, V REGISTRU A SE PREDA HODNOTA 00,
        ; NASTANE-LI CHYBA, PREDA SE 01
```

```
        ;
        ; V TOMTO PRIPADE JSOU HODNOTY CISLA DISKOVE
        ; JEDNOTKY, STOPY, SEKTORU A ADRESY DMA
        ; ULOZENY V PROMENNYCH DISKNO, TRACK,
        ; SECTOR, DMAAD
        DS      256             ; MISTO RESERVOVANE PRO
                                ; VLASTNI FYZICKE OPERACE
        LD      A,1             ; CHYBOVY STAV
        RET                     ; ZMENIT PO DOPLNENI
;
;       ZBYTEK MODULU CBIOS JE REZERVOVANA DATOVA
;       OBLAST (NEINICIALIZOVANA), KTERA NEMUSI BYT
;       SOUCASTI OBRAZU SYSTEMU NA DISKU, MUSI VSAK
;       BYT DOSTUPNA PRI BEHU SYSTEMU MEZI ADRESAMI
;       BEGDAT A ENDAT
;
TRACK:  DS      2               ; DVA BYTY PRO PRIP. ROZSIRENI
SECTOR: DS      2               ; DVA BYTY PRO PRIP. ROZSIRENI
DMAAD:  DS      2               ; ADRESA DMA
DISKNO: DS      1               ; CISLO DISKU 0, . . , 15
;
;       PRACOVNI OBLAST VYUZIVANA MODULEM BDOS
BEGDAT  EQU     $               ; ZACATEK DATOVE OBLASTI
DIRBF:  DS      128             ; PRACOVNI OBLAST NA ADRESAR
ALL00:  DS      31              ; BITOVA MAPA 0
ALL01:  DS      31              ; BITOVA MAPA 1
ALL02:  DS      31              ; BITOVA MAPA 2
ALL03   DS      31              ; BITOVA MAPA 3
CHK00.  DS      16              ; KONTROLNI VEKTOR 0
CHK01:  DS      16              ; KONTROLNI VEKTOR 1
CHK02:  DS      16              ; KONTROLNI VEKTOR 2
CHK03:  DS      16              ; KONTROLNI VEKTOR 3
;
ENDDATEQU       $               ; KONEC DATOVE OBLASTI
DATSIZ  EQU     $-BEGDAT        ; DELKA DATOVE OBLASTI
        END
```

```
        ; PROGRAM GETSYS A PUTSYS

        ; ZACATEK PROGRAMU NA POCATKU TPA
        ORG     0100H
;
MSIZE   EQU     20              ; KONFIGURACE PAMETI V KB
;       BIAS JE POSUN ADRES AKTUALNI INSTALOVANE VERZE
;       SYSTEMU CP/M VZHLEDEM K ADRESE 3400H PRO SYS-
;       TEM, KTERY OVLADA PAMETOVOU KONFIGURACI VETSI
;       NEŽ 20 KB
BIAS    EQU     (MSIZE-20)*1024
CCP     EQU     3400H+1024
BDOS    EQU     XXP+0800H
BIOS    EQU     CCP+1600H
;
;       PROGRAM GETSYS PRENASI OBSAH STOPY 0 A 1 DO
;       PAMETI OD ADRESY 3380H+BIAS
;
;       REGISTR                 POUZITI
;         A                     PRACOVNI
;         B                     CITAC STOP(0 . . . 76)
;         C                     CITAC SEKTORU
;         D,E                   PRACOVNI
;         H,L                   UKLADACI ADRESA
;         SAP                   NASTAVENI ZASOBNIKU
;
GSTART:LD       SP,CCP-0080H    ; START GETSYS
        LD      HL,CCP-0080H    ; POCATECNI ADRESA
        LD      B,0             ; STOPA 0
RD$TRK:                         ; CTI STOPU
        LD      C,1             ; SEKTOR 1
RD$SEC:                         ; CTI SEKTOR
        CALL    READ$SEC        ; FYZICKE CTENI SEKTORU
        LD      DE,128          ; POSUN O 128 BYTU
        ADD     HL,DE           ; (HL:=HL+128)
```

```
        INC     C               ; DALSI SEKTOR
        LD      A,C             ; JE-LI CISLO SEKTORU
        CP      27              ; MENSI NEZ 27
        JP      C,RD$SEC        ; CTI DALSI
;       KONEC STOPY POSUN NA DALSI
        INC     B               ; STOPA:=STOPA+1
        LD      A,B             ; JE-LI CISLO STOPY
        CP      2               ; MENSI NEZ 2
        JP      C,RD$TRK        ; CTI DALSI
;
;       KONEC CTENI SYSTEMU - HALT
;
        EI
        HALT
;
;
;       PROGRAM PUTSYS ULOZI OBRAZ SYSTEMU Z OPERACNI
;       PAMETI OD ADRESY 3380H+BIAS NA STOPY 0 A 1
;
        ORG     ($+100H) AND 0FF00H ; ZACATEK NA NOVE
                                    ; STRANCE PAMETI
PUT$SYS:
        LD      SP,CCP-0080H    ; ZASOBNIK
        LD      HL,CCP-0080H    ; POCATECNI ADRESA
        LD      B,0             ; STOPA 0
WR$TRK:
        LD      C,1             ; SEKTOR 1
WR$SEC:
        CALL    WRITE$SEC       ; FYZICKY ZAPIS SEKTORU
        LD      DE,128          ; POSUN O 128 BYTU
        ADD     HL,DE           ; (HL=HL+128)
        INC     C               ; DALSI SEKTOR
        LD      A,C             ; JE-LI CISLO SEKTORU
        CP      27              ; MENSI NEZ 27
        JP      C,WR$SEC        ; ZAPIS DALSI
```

```
;       KONEC STOPY POSUN NA DALSI
            INC     B               ; STOPA=STOPA+1
            LD      A,B             ; JE-LI CISLO STOPY
            CP      2               ; MENSI NEZ 2
            JP      C,WR$TRK        ; ZAPIS DALSI
;
;       KONEC ZAPISU SYSTEMU - HALT
;           EI
            HALT
;
;       UZIVATELEM VYTVORENE PODPROGRAMY PRO
;       FYZICKE CTENI A ZAPIS NA DISK
;
;       POSUN NA DALSI STRANKU PAMETI
            ORG     ($+100H) AND 0FF00H
READ$SEC:
            ; CTENI SEKTORU
            ; STOPA V B
            ; SEKTOR V C
            ; DMAADR V HL
            PUSH    BC
            PUSH    HL
            ; MISTO PRO UZIVATELEM DEFINOVANOU OPERACI
            DS      64
            POP     HL
            POP     BC
            RET
;
;       POSUN NA DALSI STRANKU PAMETI
            ORG     ($+100H) AND 0FF00H
WRITE$SEC:
            ; ZAPIS SEKTORU
            ; STOPA V B
            ; SEKTOR V C
            ; DMADR V HL
```

```
        PUSH    BC
        PUSH    HL
        ; MISTO PRO UZIVATELEM DEFINOVANOU OPERACI
        DS      64
        POP     HL
        POP     BC
        RET
;       KONEC PROGRAMU GETSYS/PUTSYS
        END


        ;   SYSTEMOVY ZAVADEC PRO STUDENY START
;
; TOTO JE VZOROVY ZAVADECI PROGRAM PRO STUDENY
; START. JEHO MODIFIKACE JE STANDARTNE ULOZENA V PRV-
; NIM SEKTORU STOPY 00 (PRVNI SEKTOR DISKETY - BOOT
; SEKTOR). PREDPOKLADAME, ZE TECHNICKY ZAVADEC POCI-
; TACE (PRIP. RADIC DISKU) ZAVEDL TENTO SEKTOR DO
; PAMETI PRI ZAPNUTI POCITACE (MUZE TAKE EXISTOVAT
; V PAMETI ROM, NEBO MUZE BYT VLOZEN RUCNE Z KLAVES-
; NICE). SYSTEMOVY ZAVADEC PRENESE OBRAZ SYSTEMU
; CP/M DO OPERACNI PAMETI POCINAJE ADRESOU 3400H+BIAS
; V SYSTEMU S PAMETOVOU KONFIGURACI 20 KB JE HODNOTA
; BIAS=0000H. PRI VETSI KONFIGURACI PAMETI UDAVA HOD-
; NOTA BIAS ZVETSENI PAMETI NAD 20 KB. PO VYTVORENI
; OBRAZU SYSTEMU V OPERACNI PAMETI PREDA SYSTEMOVY
; ZAVADEC RIZENI NA ZACATEK MODULU BIOS - NA VSTUPNI
; BOD PRO SLUZBU "BOOT" (STUDENY START) MODULU BIOS,
; JEHOZ ADRESA JE 3400H+BIAS+1600H. SYSTEMOVY ZAVADEC
; BUDE ZNOVU POUZIT AZ PRI DALSIM ZAPNUTI POCITACE,
; POKUD NEDOSLO K PREPSANI MODULU BIOS. POCATECNI
; ADRESA VZOROVEHO SYSTEMOVEHO ZAVADECE JE 0000H A
; MUSI BYT ZMENENA, JESTLIZE TECHNICKY ZAVADEC ZAVE-
; DE PRVNI SEKTOR JINAM, NEBO JE-LI POUZITA PAMET ROM.
;
```

```
        ORG     0
MSIZE   EQU     20                  ; MIN. DELKA PAMETI V KB
BIAS    EQU     (MSIZE-20)*1024     ; POSUN PRO VETSI
                                    ; KONFIGURACI
CCP     EQU     3400H+BIAS          ; POCATEK MODULU CCP
BIOS    EQU     CCP+1600H           ; POCATEK MODULU BIOS
BIOSL   EQU     0300H               ; DELKA MODULU BIOS
BOOT    EQU     BIOS                ; VSTUPNI BOD SLUZBY BOOT
SIZE    EQU     BIOS+BIOSL - CCP ; DELKA SYSTEMU CP/M
SECTS   EQU     SIZE/128    ; DELKA SYSTEMU V SEKTORECH
;
COLD:   ; VSTUPNI BOD SYSTEMOVEHO ZAVADECE
;
        LD      BC,2                ; B:=0, C:=SEKTOR 2
        LD      D,SECTS             ; POCET SEKTORU, KTERE SE
                                    ; MAJI CIST
        LD      HL,CCP              ; BAZE PRO PRENOS
;
LSECT:  ; ZAVED DALSI SEKTOR
;
;       MISTO NA VLOZENI PROGRAMU, KTERY PRECTE JEDEN
;       SEKTOR DELKY 128 SLABIK ZE STOPY DANE REGISTREM
;       B, CISLO SEKTORU JE V REGISTRU C, NA ADRESU
;       DANOU OBSAHEM REGISTROVEHO PARU HL
;
; NASTANE-LI CHYBA PRI CTENI, SKOCI SE NA ADRESU COLD
; (ZAVEDENI SE OPAKUJE)
;
;*****************************************************************
;*       UZIVATELEM DEFINOVANA OPERACE CTENI                     *
;*****************************************************************
;
        JP      PAST$PATCH      ; TUTO INSTRUKCI JE NUTNO
                                ; ODSTRANIT PO DOPLNENI
                                ; PROGRAMU PRO CTENI
```

```
        DS      60H
PAST$PATCH:
;
; PO PRENOSU JEDNOHO SEKTORU PRECHOD NA DALSI SEK-
; TOR. POKUD JSOU JIZ PRENESENY VSECHNY SEKTORY OBRA-
; ZU SYSTEMU, AKTIVUJE SE SLUZBA BOOT MODULU BIOS.
;
        DEC     D               ; SECTS:=SECTS-1
        JP      Z,BOOT          ; VOLANI SLUZBY BOOT
;
; ZBYVA PRENEST DALSI SEKTORY
; (NEPOUZIVAME ZDE ZASOBNIK, PROTO MUZEME REGISTR SP
; POUZIT JAKO PRACOVNI REGISTR PRO USCHOVANI DELKY
; SEKTORU, KTEROU PRICITAME K ADRESE PRO PRENOS)
;
        LD      SP,128          ; 128 BYTU NA SEKTOR
        ADD     HL,SP           ; HL:=HL+128
;
; PRECHOD NA DALSI SEKTOR. POKUD BYL PRENESEN
; POSLEDNI SEKTOR NA STOPE, PRECHOD NA PRVY SEKTOR
; NASLEDUJICI STOPY.
;
        INC     C               ; SEKTOR:=SEKTOR+1
        LD      A,C
        CP      27
        JP      C,LSECT         ; NENI POSLEDNI —> DALSI
        LD      C,1             ; SEKTOR:=1
        INC     B               ; STOPA:=STOPA+1
        JP      LSECT           ; DALSI ZAVADENI
        END
```

Generování úrovně 2

Generování úrovně 2 se skládá z *rekonfigurace systému* pro požadovanou velikost operační paměti a z *rozšíření modulu* BIOS. Přitom máme k dispozici fungující operační systém (tím se úroveň 2 liší od úrovně 1, kterou je zpravidla nutno provádět na jiném počítači, příp. pomocí jiného systému). Generování úrovně 2 sestává z následujících kroků.

1. Pomocí textového editoru a překladače vytvoříme konečnou verzi modulu BIOS. Doplníme obsluhu všech diskových jednotek, v případě potřeby blokovací algoritmus, ošetření chyb, obsluhu znakových periférií apod. Při rozšiřování modulu BIOS (pokud je modul BIOS zaváděn z disku) může nastat případ, že se celý modul BIOS nevejde na vyhrazené místo v systémové oblasti disku (standardně 380H slabik v systémové oblasti na první stopě). V takovém případě je možno postupovat např. následovně:
– Rezervovat větší počet stop pro systémovou oblast (např. 3 stopy). Nevýhodou je ztráta kompatibility pro přenos souborů, neboť systém pak předpokládá, že adresář je uložen na jiném místě než na standardních disketách.
– Umístit část modulu BIOS jinde (např. na poslední stopě, ve speciálním souboru apod.)

2. Rekonfigurace modulů CCP a BDOS. Provádí se pomocí programu MOVCPM příkazem

A>MOVCPM xx *,

kde xx je velikost paměti dekadicky v KB. (Velikost paměti je vztažena k adrese BIOS+600H).

Verze programu MOVCPM musí souhlasit s verzí operačního systému, ve které je provozován. V opačném případě program MOVCPM skončí činnost chybovou zprávou (SYNCHRONIZATION ERROR). Při korektní činnosti vypíše program MOVCPM zprávu:

CONSTRUCTING xxK CP/M VERS 2.n
READY FOR "SYSGEN" OR
"SAVE 34 CPMxx.COM"

a skončí. V tomto okamžiku je v operační paměti od adresy 900H uložena rekonfigurovaná verze systému (980–117FH modul CCP, 1180H–1F7FH

modul BDOS). Uvedeným příkazem SAVE je třeba uložit tuto verzi do souboru na disku.

3. Pomocí programu DDT se doplní k rekonfigurovaným modulům CCP a BDOS upravený modul BIOS a systémový zavaděč, např.

```
A>DDTxx.COM    (zavede rekonfigurovanou verzi)
-IOBIOS.HEX    (připojí modul BIOS od adresy 1F80H)
-Rn,
```

kde n=1F80H–BIOS (lze určit příkazem H).

Obdobným způsobem se v případě potřeby doplní systémový zavaděč na adresu 900H. Poté se program DDT ukončí (G0 nebo ^C). Upravenou verzi lze příkazem SAVE uložit do souboru na disk.

4. Nahrání obrazu systému z operační paměti do systémové oblasti disku provádí služební program SYSGEN (v podstatě se jedná o spojené programy GETSYS a PUTSYS). Vyvolání programu SYSGEN je standardní:

```
A>sysgen
SYSGEN VERSION 2.x
SOURCE DRIVE NAME (OR RETURN TO SKIP)
```

Na tuto výzvu odpovíme stiskem klávesy RETURN, neboť obraz systému je již vytvořen v operační paměti. Komunikace pokračuje výzvou:

```
DESTINATION DRIVE NAME (OR RETURN TO SKIP) n
```

Zde je třeba odpovědět označením jednotky, na které budeme nahrávat (zde symbolicky n). Program SYSGEN reaguje výzvou:

```
DESTINATION ON n, THEN TYPE RETURN
```

pro založení pracovní diskety do vybrané jednotky a stisk klávesy RETURN. Program SYSGEN nahraje obsah operační paměti do systémové oblasti na disku a ohlásí provedení funkce zprávou:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)
```

Činnost programu SYSGEN se zakončí stiskem klávesy RETURN.

Vytvořená disketa se založí do systémové jednotky (A:) a stiskem ^C se provede teplý start. Je-li vše v pořádku, systém se ohlásí standardní výzvou A> a je možno jej otestovat.

Poznámka: v distribuční verzi systému CP/M jsou definovány konstanty OFFSET a BIAS, jejichž význam je následující:

– OFFSET je hodnota, o kterou je nutno zvětšit každou adresu v aktuální (živé) verzi systému CP/M abychom dostali (modulo 64 KB) odpovídající adresu v distribuční verzi, tj. v obrazu systému, který začíná na adrese 900H. Pokud je např. aktuální modul BIOS uložen od adresy 0FA00H (tj. celková délka modulu BIOS je 600H), zatímco v generační verzi je modul BIOS uložen na adrese 1F80H, je hodnota OFFSET určena rovností:

0FA00H + OFFSET = 1F80H,

tj. OFFSET=2580H.

– BIAS je hodnota, o kterou je nutno zvětšit adresy v distribuční verzi systému CP/M (verzi pro 20 KB), abychom získali adresu v aktuální verzi systému. Je-li např. aktuální modul BIOS umístěn na adrese 0FA00H, je hodnota konstanty BIAS=0B000H (tj. 44 KB), neboť v distribuční verzi pro 20 KB je modul BIOS umístěn od adresy 4A00H (tj. 5000H – 600H). Pak platí:

4A00H + BIAS = 4A00H + 0B000H = 0FA00H.

7 Přílohy

7.1 Tabulky kódu ASCII

Tab. 7.1 udává přehled *hexadecimálních číselných kódů* znaků v kódu ASCII (American Standard Code for Information Interchange), tj. kódu KOI 7. Kód je uveden s potlačenou paritou, tj. nejvyšší (paritní) bit je nulový.

Tab. 7.1. Hexadecimální hodnoty ASCII kódu

00	NUL	10	DLE	20	SPA	30	0	40	@	50	P	60	‘	70	P
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	3	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	46	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	ETB	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	n	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	-	6F	o	7F	DEL

Tab. 7.2 udává přehled *dekadických číselných kódů* znaků v kódu ASCII (American Standard Code for Information Interchange), tj. kódu KOI 7. Kód je uveden s potlačenou paritou, tj. nejvyšší (paritní) bit je nulový.

Tab. 7.2. Dekadické hodnoty ASCII kódu

000	NUL	016	DLE	032	SPA	048	0	064	@	080	P	096	‘	112	p
001	SOH	017	DC1	033	!	049	1	065	A	081	Q	097	a	113	q
002	STX	018	DC2	034	"	050	2	066	B	082	R	098	b	114	r
003	ETX	019	DC3	035	#	051	3	067	C	083	S	099	c	115	s
004	EOT	020	DC4	036	$	052	4	068	D	084	T	100	d	116	t
005	ENQ	021	NAK	037	%	053	5	069	E	085	U	101	e	117	u
006	ACK	022	SYN	038	&	054	6	070	F	086	V	102	f	118	v
007	BEL	023	ETB	039	’	055	7	071	G	087	W	103	g	119	w
008	BS	024	ETB	040	(	056	8	072	H	088	X	104	h	120	x
009	HT	025	EM	041	)	057	9	073	I	089	Y	105	i	121	y
010	LF	026	SUB	042	*	058	:	074	J	090	Z	106	j	122	z
011	VT	027	ESC	043	+	059	’	075	K	091	[	107	k	123	{
012	FF	028	FS	044	,	060	<	076	L	092	\	108	l	124	|
013	CR	029	GS	045	-	061	=	077	M	093	]	109	m	125	}
014	SO	030	RS	046	.	062	>	078	N	094	^	110	n	126	~
015	SI	031	US	047	/	063	?	079	O	095	-	111	o	127	DEL

Tab. 7.3 udává přehled *oktalových číselných kódů* znaků v kódu ASCII (American Standard Code for Information Interchange), tj. kódu KOI 7. Kód je uveden s potlačenou paritou, tj. nejvyšší (paritní) bit je nulový.

Tab. 7.3. Oktalové hodnoty ASCII kódu

000	NUL	020	LE	040	SPA	060	0	100	@	120	P	140	‘	160	p
001	SOH	021	DC1	041	!	061	1	101	A	121	Q	141	a	161	q
002	STX	022	DC2	042	"	062	2	102	B	122	R	142	b	162	r
003	ETX	023	DC3	043	#	063	3	103	C	123	S	143	c	163	s
004	EOT	024	DC4	044	$	064	4	104	D	124	T	144	d	164	t
005	ENQ	025	NAK	045	%	065	5	105	E	125	U	145	e	165	u
006	ACK	026	SYN	046	&	066	6	106	F	126	V	146	f	166	v
007	BEL	027	ETB	047	’	067	7	107	G	127	W	147	g	167	w
010	BS	030	ETB	050	(	070	8	110	H	130	X	150	h	170	x
011	HT	031	EM	051	)	071	9	111	I	131	Y	151	i	171	y
012	LF	032	SUB	052	*	072	:	112	J	132	Z	152	j	172	z
013	VT	033	ESC	053	+	073	;	113	K	133	[	153	k	173	{
014	FF	034	FS	054	,	074	<	114	L	134	\	154	l	174	|
015	CR	035	GS	055	-	075	=	115	M	135	]	155	m	175	}
016	SO	036	RS	056	.	076	>	116	N	136	^	156	n	176	~
017	SI	037	US	057	/	077	?	117	O	137	-	157	o	177	DEL

7.2 Chybové zprávy systému CP/M

Následující seznam chybových zpráv obsahuje zprávy standardního systému CP/M, včetně služebních programů. Neobsahuje žádné zprávy modulu BIOS, neboť ten je závislý na konkrétní instalaci a jeho zprávy na jeho tvůrci.

Chybové zprávy jsou seřazeny podle abecedy a jsou uváděny bez syntaktických příkras (např. místo: "** Aborted **" jen "ABORTED") a pouze velkými písmeny (byť se v konkrétní instalaci mohou zobrazovat kombinací malých a velkých písmen).

Za každou chybovou zprávou je uveden (v závorkách) program, který ji hlásí, spolu s popisem možných příčin..

? . . . (CCP)
– Neznámý příkaz pro CCP – snaha spustit program "name", pro nějž na disku neexistuje soubor "name.COM".
– Chybí parametr počet stránek v příkazu SAVE.

? . . . (DDT)
– DDT nemůže najít udaný soubor.
– Chyba v souboru typu HEX (chybný kontrolní součet, chyba v položce).
– Chybná instrukce v příkazu A (Assemble) (chybný operační kód, znak H za zápisem hexadecimálního čísla apod.).

?? . . . (DDT)
– Při výpisu obsahu paměti příkazem L (List – inverzní asembler) byl nalezen neznámý kód instrukce pro procesor 8080 (např. při výpisu programu pro procesor Z80 či zóny obsahující text).

ABORTED . . . (STAT)
– Přerušení činnosti STAT (stiskem libovolné klávesy) během nastavování atributů souborů ($SYS, $DIR, $R/W, $R/O). V jiných případech STAT nelze přerušit.

ABORTED . . . (PIP)
– Přerušení činnosti PIP (stiskem libolné klávesy) během kopírování souboru na logické zařízení pro tisk (LST:).

BAD DELIMITER . . . (STAT)
– Chybný omezovač v parametrech STAT při nastavování přiřazení fyzických zařízení logickým (STAT log:=fyz:).

BAD LOAD . . . (CCP)
– Příliš velký program – snaha zavést pomocí CCP program, jehož délka je větší než oblast programů (TPA).

BAD PARAMETER . . . (PIP)
– Chybný přepínač pro PIP (neznámý přepínač, chybná hodnota přepínače).

BDOS ERROR ON d: BAD SECTOR . . . (BDOS)
– Chybné čtení nebo zápis na diskové jednotce d:. BDOS hlásí tuto chybu v případě, kdy mu funkce READ nebo WRITE modulu BIOS vrátí nenulovou hodnotu v registru A jako indikaci chyby. Při odpovědi ^C je znovu zaveden systém (vyvolána funkce WBOOT), při stisku CR (RETURN apod.) je chyba ignorována – pozor na případné důsledky! Správně vytvořený modul BIOS obvykle nepovolí uživateli ignorování chyby (ošetří zpracování chyby sám). Doporučujeme používání odpovědi ^C.

BDOS ERROR ON d: FILE R/O . . . (BDOS)
– Pokus o zrušení souboru s atributem $R/O na diskové jednotce d:. BDOS čeká na stisk libovolné klávesy a poté znovu zavede systém (vyvolá funkci WBOOT modulu BIOS). Vzhledem k možným nejednoznačnostem při použití označení skupiny souborů (např.: ERA *.typ), doporučujeme použití služebního programu STAT k analýze výsledného stavu a příčiny této chyby.

BDOS ERROR ON d: R/O . . . (BDOS)
– Pokus o zápis na diskovou jednotku d:, která má status $R/O. Obvykle následek výměny diskového média bez následného stisku ^C. BDOS čeká na stisk libovolné klávesy a poté znovu zavede systém (vyvolá funkci WBOOT modulu BIOS).

BDOS ERROR ON d: SELECT . . . (BDOS)
– Pokus o přístup na neexistující logickou diskovou jednotku. BDOS hlásí tuto chybu v případě, kdy mu funkce SELDSK vrátí adresu 0 ve dvojici registrů HL jako indikaci neexistence potřebných tabulek pro nastavovaný

disk. V případě, že tato chyba byla způsobena chybným příkazem pro nastavení aktuální diskové jednotky, nezbývá než fyzicky znovu zavést systém (stisk RESET apod.). Pokud byla tato chyba způsobena pokusem vykonat příkaz (program) odkazující se na neexistující diskovou jednotku, je možno stiskem ^C vyvolat teplý start systému (BDOS vyvolá funkci WBOOT modulu BIOS).

BREAK x AT y . . (ED)
– Chyba při běhu programu ED (y je číslo řádku, při jehož zpracování k chybě došlo):

x = "#" – požadovaný řetěz nebyl nalezen,
x = "?" – neznámý příkaz pro ED,
x = "0" – požadovaný soubor nebyl nalezen,
x = ">" – přeplnění vnitřní paměti ED,
x = "E" – zpracování příkazu bylo přerušeno,
x = "F" – plný disk nebo adresář.

CANNOT CLOSE, READ/ONLY? . . . (SUBMIT)
– Chyba při vytváření dávky "$$$.SUB" programem SUBMIT: disk na nějž má být dávka zapsána je fyzicky chráněn proti zápisu (nesouvisí s logickou ochranou diskové jednotky).

CANNOT CLOSE DESTINATION FILE . . . (PIP)
– Chyba při výstupu na disk programem PIP. Možnou příčinou je disk fyzicky chráněný proti zápisu. Pokud není disk fyzicky chráněn proti zápisu, došlo k chybě technického rázu.
– Položka adresáře byla zapsána na chybné místo nebo je chybný disk apod.

CANNOT CLOSE FILES . . . (ASM)
– Chyba při výstupu na disk programem ASM. Možnou příčinou je disk fyzicky chráněný proti zápisu. Pokud není disk fyzicky chráněn proti zápisu, došlo k chybě technického rázu.
– Položka adresáře byla zapsána na chybné místo nebo je chybný disk apod.

CANNOT READ . . . (PIP)
– Chyba v parametrech programu PIP. Pokus o čtení z logického zařízení, které je určeno pouze pro výstup (např.: PIP name.typ=LST:).

CANNOT WRITE . . . (PIP)
– Chyba v parametrech PIP. Pokus o zápis na logické zařízení, které je určeno pouze pro vstup (např.: PIP RDR:=name.typ).

CHECKSUM ERROR . . . (LOAD)
– Ve vstupním souboru (typu HEX) pro program LOAD byl zjištěn chybný kontrolní součet řádku. Program LOAD dále zobrazí informaci o místě vzniku chyby ve tvaru:

```
CHECKSUM ERROR
LOAD ADDRESS 0210        <— adresa začátku řádku
ERROR ADDRESS 0212       <— adresa chyby
BYTES READ:
0210: 00 33 22 2B 02 21 27 02
```

CHECKSUM ERROR . . . (PIP)
– Při kopírování souboru typu HEX programem PIP byl ve vstupním souboru indikován chybný kontrolní součet řádku (viz přepínače [H] a [I] programu PIP).

COMMAND BUFFER OVERFLOW . . . (SUBMIT)
– Soubor obsahující dávku příkazů zpracovávaný programem SUBMIT je příliš dlouhý – maximální délka příkazové dávky je 2048 znaků (délka vnitřní paměti SUBMIT). Příslušnou dávku je třeba zkrátit nebo rozdělit na dávky dvě (viz popis SUBMIT).

COMMAND TOO LONG . . . (SUBMIT)
– Příliš dlouhý řádek čtený programem SUBMIT. Maximální délka příkazu (příkazového řádku) čteného z dávky je 125 znaků.

CORRECT ERROR, TYPE RETURN OR CTL–Z . . . (PIP)
– Opravitelná chyba při snímání děrné pásky (např. indikace konce pásky). Stiskem ^Z se snímání ukončí, po stisku RETURN (CR, ENTER atd.) PIP pokračuje ve čtení.

DESTINATION IS R/O, DELETE (Y/N)? . . . (PIP)
– Program PIP si žádá potvrzení pro přepsání souboru, který má atribut $R/O (viz popis STAT). Potvrdíme-li žádost stiskem Y, přepíše PIP soubor

a ponechá jeho atributy beze změny. Stiskem libovolné jiné klávesy zabráníme přepsání souboru a PIP vypíše zprávu:

** NOT DELETED **

Použijeme-li přepínač [W] v příkazu pro PIP, je soubor přepsán bez vyžádání potvrzení.

DIRECTORY FULL . . . (SUBMIT)
– Program SUBMIT nemůže vytvořit dávku příkazů "$$$.SUB", protože není místo v adresáři. Může být způsobeno též chybně formátovaným diskem (položky adresáře nezačínají 0E5H) – lze zjistit pomocí STAT USR:, který vypíše uživatele s čísly různými od povolených čísel 0 ... 15.

DISK OR DIRECTORY FULL . . . (ED)
– Program ED nemůže vytvořit výstupní soubor z důvodu vyčerpání kapacity disku nebo přeplnění adresáře.

DISK READ ERROR . . . (PIP)
– Chyba při čtení z disku programem PIP. Obvykle ji předchází hlášení modulu BIOS, který by měl situaci řešit sám.

DISK WRITE ERROR . . . (PIP)
– Chyba při zápisu na disk programem PIP. Obvykle ji předchází hlášení modulu BIOS, který by měl situaci řešit sám.

DISK WRITE ERROR . . . (SUBMIT)
– Chyba při zápisu na disk programem SUBMIT. Obvykle ji předchází hlášení modulu BIOS, který by měl situaci řešit sám.

END OF FILE, CTL–Z? . . . (PIP)
– Indikace nalezení znaku ^Z při kopírování souboru typu HEX programem PIP. Slouží k možnému skládání více souborů typu HEX (např. z více děrných pásek). Odpověď ^Z způsobí ukončení vstupu; odpovíme-li libovolným jiným znakem, pokračuje PIP v kopírování.

ERROR: CANNOT CLOSE FILES . . . (LOAD)
– Program LOAD zjistil, že výstupní disk je fyzicky chráněn proti zápisu.

ERROR: CANNOT OPEN SOURCE . . . (LOAD)
– Program LOAD nenalezl soubor typu HEX uvedený jako parametr.

ERROR: DISK READ . . . (LOAD)
– Chyba při čtení z disku programem LOAD. Obvykle ji předchází hlášení modulu BIOS, který by měl situaci řešit sám.

EROR: DISK WRITE . . . (LOAD)
– Chyba při zápisu na disk programem LOAD. Obvykle ji předchází hlášení modulu BIOS, který by měl situaci řešit sám.

ERROR: INVERTED LOAD ADDRESS . . . (LOAD)
– Program LOAD zjistil, že zaváděcí adresa ve vstupním souboru typu HEX je menší než 0100H. (Pozn.: Program DDT stejný soubor zpracuje bez hlášení chyby, ale s příslušným efektem na komunikační zónu.)

ERROR: NO MORE DIRECTORY SPACE . . . (LOAD)
– Program LOAD nemůže zapsat výstupní soubor, protože je plný adresář disku.

ERROR ON LINE n . . . (SUBMIT)
– Program SUBMIT neumí zpracovat n-tý řádek ve vstupním souboru typu SUB (pravděpodobně tento soubor neobsahuje text).

FILE EXISTS . . . (CCP)
– Chyba v příkazu REN – nové jméno souboru je v kolizi s již existujícím souborem (soubor stejného jména již existuje). Způsobí ignorování příkazu REN.

FILE IS READ/ONLY . . . (ED)
– Pokus o editaci souboru s atributem $R/O programem ED.

FILE NOT FOUND . . . (STAT)
– Neexistující soubor v parametrech programu STAT.

FILENAME NOT FOUND . . . (PIP)
– Neexistující soubor v parametrech programu PIP.

INVALID ASSIGNMENT . . . (STAT)
– Chyba v parametrech programu STAT pro přiřazení fyzických zařízení logickým. Doporučujeme použít STAT VAL: ke zjištění chyby.

INVALID CONTROL CHARACTER . . . (SUBMIT)
– Chybný řídicí znak (^x) ve vstupní dávce. (Pozn.: Starší verze SUBMIT nedovolují použít řídicích znaků.) Pravděpodobně výskyt kombinace ^x, kde x je různé od "A" . . "Z".

INVALID DIGIT . . . (PIP)
– Chyba ve vstupu programu PIP – v místě, kde jsou očekávána numerická data, se vyskytl nenumerický znak.

INVALID DISK ASSIGNMENT . . . (STAT)
– Chyba v parametrech programu STAT pro pracovní nastavení zákazu zápisu na disk (STAT d:=R/O – obvykle znak $ před R/O, který se používá u nastavení atributů souborů). Doporučujeme použít STAT VAL: ke zjištění správného tvaru.

INVALID DRIVE NAME (USE A, B, C, OR D) . . . (SYSGEN)
– Pokus o čtení či zápis systému na jinou diskovou jednotku než A, B, C nebo D (SYSGEN umí pracovat pouze s těmito jednotkami).

INVALID FILE INDICATOR . . . (STAT)
– Neznámý typ atributu souboru v parametrech programu STAT. Povolené atributy jsou:

$SYS – systémový soubor,
$DIR – běžný soubor,
$R/O – soubor chráněný proti zápisu,
$R/W – soubor lze přepisovat.

INVALID FORMAT . . . (PIP)
– Chyba v parametrech programu PIP (např. "-" místo "=").

INVALID HEX DIGIT . . . (LOAD)
– Program LOAD nalezl ve vstupním souboru znak neodpovídající hexadecimální číslici. Dále je zobrazena informace:

INVALID HEX DIGIT
LOAD ADDRESS 0210 <— adresa začátku řádku
ERROR ADDRESS 0212 <— adresa chyby
BYTES READ:
0210: 00 33

INVALID MEMORY SIZE . . . (MOVCP)
– Chybně zadaná velikost paměti při rekonfiguraci systému programem MOVCPM.

INVALID SEPARATOR . . . (PIP)
– Chybný oddělovač vstupních souborů v parametrech programu PIP (musí být výhradně " , ").

INVALID USER NUMBER . . . (PIP)
– Chybné číslo uživatele (jiné než 0, ... , 15) v přepínači [Gn] v parametrech programu PIP.

NO 'SUB' FILE PRESENT . . . (SUBMIT)
– Program SUBMIT nemůže nalézt soubor uvedený v parametrech jako jméno dávky (tj. např. pro SUBMIT JOB neexistuje na aktuální diskové jednotce soubor JOB.SUB).

NO DIRECTORY SPACE . . . (ASM)
– Program ASM nemůže založit výstupní soubor, protože je plný adresář disku.

NO DIRECTORY SPACE . . . (PIP)
– Program PIP nemůže založit výstupní soubor, protože je plný adresář disku.

NO FILE . . . (CCP)
– Chyba v příkazu REN – soubor, který se má přejmenovat na disku, neexistuje.
– Zpráva při požadavku na výpis adresáře prázdné skupiny souborů (prázdného disku).

NO FILE . . . (PIP)
– Program PIP nemůže najít vstupní soubor uvedený v parametrech.

NO MEMORY . . . (ED)
– Program ED nemá dostatek paměti pro úschovu editovaného souboru.

NO SOURCE FILE ON DISK . . . (SYSGEN)
– Program SYSGEN nenalezl vstupní soubor.

NO SOURCE FILE PRESENT . . . (ASM)
– Program ASM nemůže najít vstupní soubor. Je nutno si uvědomit, že parametry se ASM předávají na místě typu souboru, který naopak musí být povinně ASM (např. ASM SOURCE.AAA neznamená, že vstupní soubor je SOURCE.AAA; vstupním souborem je soubor SOURCE.ASM, výstupními soubory SOURCE.HEX a SOURCE.PRN).

NO SPACE . . . (CCP)
– Chyba při zpracování příkazu SAVE, kdy na čisku není dostatek místa pro uložení daného obrazu paměti.

NOT A CHARACTER SOURCE . . . (PIP)
– Pokus o vstup znaků ze zařízení, které to neumožňuje (např. PIP name=PUN:).

OUTPUT FILE WRITE ERROR . . . (ASM)
– Program ASM hlásí tuto zprávu, pokud mu BDOS hlásí chybu při zápisu na disk. (Obvykle ošetřeno v rámci modulu BIOS).

PARAMETER ERROR . . . (SUBMIT)
– V popisu dávky příkazů zpracovávaném programem SUBMIT se vyskytl chybný odkaz na parametr. Povolená označení míst pro substituci parametrů jsou $1, ... , $9. Chybu pravděpodobně způsobil abecední znak bezprostředně za znakem $$; např. namísto $$$.SUB musíme psát $$$$$$.SUB.

PERMANENT ERROR, TYPE RETURN TO IGNORE . . . (SYSGEN)
– Program SYSGEN hlásí tuto chybu, pokud mu modul BIOS vrátí indikaci chyby při čtení nebo zápisu na disk. (Obvykle řeší tuto situaci BIOS sám).

QUIT NOT FOUND ... (PIP)
– Zpráva programu PIP, pokud nebyl nalezen řetěz string při použití přepínače [Qstring^Z] (znamená přerušení výstupu při nalezení řetězu string).

READ ERROR ... (CCP)
– Tuto chybu hlásí CCP, pokud mu modul BIOS předá indikaci o chybě při čtení z disku (obvykle řeší tuto situaci modul BIOS sám).

RECORD TOO LONG ... (PIP)
– Program PIP nalezl ve vstupním souboru typu HEX řádek delší než 80 znaků (viz přepínač [H]).

REQUIRES CP/M 2.0 OR NEWER FOR OPERATION ... (PIP)
– PIP nepracuje pod verzí systému nižší než 2.0.

REQUIRES CP/M VERSION 2.0 OR LATER ... (XSUB)
– XSUB nepracuje pod verzí systému nižší než 2.0.

SOURCE FILE INCOMPLETE ... (SYSGEN)
– Příliš krátký soubor pro SYSGEN.

SOURCE FILE NAME ERROR ... (ASM)
– V parametrech programu ASM bylo použito označení skupiny (jméno obsahující znaky "*" nebo "?").

SOURCE FILE READ ERROR ... (ASM)
– Chyba ve vstupním souboru pro ASM.

START NOT FOUND ... (PIP)
– Zpráva programu PIP, pokud nebyl nalezen řetěz string při použití přepínače [Sstring^Z] (znamená obnovení výstupu při nalezení řetězu string).

SYMBOL TABLE OVERFLOW ... (ASM)
– Příliš mnoho symbolů ve zdrojovém programu. (Program nutno rozdělit do několika sekcí s použitím pseudoinstrukce ORG).

SYNCHRONIZATION ERROR ... (MOVCPM)
– Interní číslo verze MOVCPM nesouhlasí s interním číslem verze systému, na němž MOVCPM běží.

SYSTEM FILE NOT ACCESSIBLE ... (ED)
– Pokus o editaci souboru s atributem $SYS.

TOO MANY FILES ... (STAT)
– Programu STAT nestačí vnitřní paměť pro setřídění výpisu informací o souborech. Nutno vypsat po menších skupinách.

UNRECOGNIZED DESTINATION ... (PIP)
– Chybné výstupní zařízení v parametrech programu PIP.

VERIFY ERROR ... (PIP)
– Neshoda při verifikaci informace zapsané na disk proti paměti (přepínač [V] v parametrech programu PIP). Může být způsobeno chybou zápisu na disk, chybou při čtení z disku nebo chybou paměti.

WRONG CP/M VERSION (REQUIRES 2.0) ... (STAT)
– STAT nepracuje pod verzí systému nižší než 2.0.

(XSUB ACTIVE) ... (XSUB)
– Zpráva, kterou vypisuje XSUB při každém teplém startu systému (WBOOT). Informuje uživatele, že je nastaveno předávání znaků programům z popisu dávky namísto z klávesnice (viz popis SUBMIT a XSUB).

XSUB ALREADY PRESENT ... (XSUB)
– Pokus o spuštění XSUB v okamžiku, kdy je již aktivní.

Číslo služby	Význam
0	Inicializace systému
1	Vstup znaku ze systémové konzole (CON:)
2	Výstup znaku na systémovou konzoli (CON:)
3	Vstup znaku z logického zařízení pro vstup (RDR:)
4	Výstup znaku na logické zařízení pro výstup (PUN:)
5	Výstup znaku na logické zařízení pro tisk (LST:)
6	Přímý vstup nebo výstup na systémovou konzoli
7	Dodání aktuální hodnoty slabiky IOBYTE
8	Nastavení aktuální hodnoty slabiky IOBYTE
9	Výstup řetězu znaků na systémovou konzoli
10	Čtení editovaného řádku ze systémové konzole
11	Test stavu systémové konzole
12	Dodání čísla verze systému
13	Inicializace diskového systému
14	Nastavení aktuální diskové jednotky
15	Otevření souboru
16	Uzavření souboru
17	Vyhledání prvního výskytu souboru
18	Vyhledání dalšího výskytu souboru
19	Zrušení souboru
20	Sekvenční čtení věty
21	Sekvenční zápis věty
22	Vytvoření souboru
23	Přejmenování souboru
24	Dodání vektoru aktivních diskových jednotek
25	Dodání čísla aktuální diskové jednotky
26	Nastavení adresy DMA
27	Dodání adresy bitové mapy
28	Zákaz zápisu na aktuální diskovou jednotku
29	Dodání vektoru diskových jednotek se zákazem zápisu

30	Nastavení atributů souboru
31	Dodání adresy tabulky popisu disku DPB
32	Nastavení, příp. dodání čísla aktuálního uživatele
33	Čtení věty s přímým přístupem
34	Zápis věty s přímým přístupem
35	Výpočet velikosti souboru
36	Nastavení čísla věty pro přímý přístup
37	Inicializace vybrané diskové jednotky
40	Zápis věty s přímým přístupem s doplněním nul

7.4 Přehled služeb modulu BIOS

Číslo služby	Označení služby	Význam
0	BOOT	Studený start systému
1	WBOOT	Teplý start systému
2	CONST	Zjištění stavu konzole
3	CONIN	Vstup znaku z konzole
4	CONOUT	Výstup znaku na konzoli
5	LIST	Výstup znaku na tiskárnu
6	PUNCH	Výstup znaku na děrovač
7	READER	Vstup znaku ze spínače
8	HOME	Nastavení na stopu 0
9	SELDSK	Nastavení diskové jednotky
10	SETTRK	Nastavení stopy
11	SETSEC	Nastavení sektoru
12	SETDMA	Nastavení adresy DMA
13	READ	Čtení sektoru
14	WRITE	Zápis sektoru
15	LISTST	Zjištění stavu tiskárny
16	SECTRAN	Překlad čísel sektorů

7.5 Struktura formátu HEX

Formát HEX sloužil původně k záznamu binární informace na děrnou pásku. Aby byl záznam informace čitelný, byl kódován znakově ve formě hexadecimálního výpisu binární informace. Obsah informace je rozložen do bloků, které jsou opatřeny služebními informacemi: typem bloku, délkou, zaváděcí adresou a kontrolním součtem.

Označení	Rel. adr.	Význam
RECORD MARK	0	Začátek bloku (znak " : ", 3AH)
RECORD LENGTH	1	Délka dat ve slabikách (0=eof)
LOAD ADDRESS	3	Zaváděcí (počáteční) adresa
RECORD TYPE	5	Typ bloku (0=data, 0, 1=eof)
DATA	6	Hexadecimální data (každá slabika je vyjádřena dvojicí znaků "00"–"FF")
CHECKSUM	6+n	Kontrolní součet (záporný součet modulo 256 všech slabik od RECORD LENGTH po poslední slabiku položky DATA)

Pozn.: Součet všech položek (modulo 256) včetně kontrolního součtu je 0.

7.6 Tabulka instrukcí procesorů 8080 a Z80

Tab. 7.4. Instrukční kódy procesorů 8080 a Z80

Symbolická označení instrukcí		Strojový kód			
8080	Z80	Hex	Bin	Okt	Dec
ACI data	ADC A, data	CE	11001110	316	206
ADC B	ADC A,B	88	10001000	210	136
ADC C	ADC A,C	89	10001001	211	137
ADC D	ADC A,D	8A	10001010	212	138
ADC E	ADC A,E	8B	10001011	213	139
ADC H	ADC A,H	8C	10001100	214	140
ADC L	ADC A,L	8D	10001101	215	141
ADC M	ADC A,(HL)	8E	10001110	216	142
ADC A	ADC A,A	8F	10001111	217	143
ADI data	ADD A, data	C6	11000110	306	198
ADD B	ADD A,B	80	10000000	200	128
ADD C	ADD A,C	81	10000001	201	129
ADD D	ADD A,D	82	10000010	202	130
ADD E	ADD A,E	83	10000011	203	131
ADD H	ADD A,H	84	10000100	204	132
ADD L	ADD A.L	85	10000101	205	133
ADD M	ADD A,(HL)	86	10000110	206	134
ADD A	ADD A,A	87	10000111	207	135
DAD B	ADD HL,BC	09	00001001	011	009
DAD D	ADD HL,DE	19	00011001	031	025
DAD H	ADD HL,HL	29	00101001	051	041
DAD SP	ADD HL,SP	39	00111001	071	057
ANI data	AND data	E6	11100110	346	230
ANA B	AND B	A0	10100000	240	160
ANA C	AND C	A1	10100001	241	161
ANA D	AND D	A2	10100010	242	162
ANA E	AND E	A3	10100011	243	163

ANA H	AND H	A4	10100100	244	164
ANA L	AND L	A5	10100101	245	165
ANA M	AND (HL)	A6	10100110	246	166
ANA A	AND A	A7	10100111	247	167
CALL addr	CALL addr	CD	11001101	315	205
CC addr	CALL A,addr	DC	11011100	334	220
CM addr	CALL M,addr	FC	11111100	374	252
CNC addr	CALL NC,addr	D4	11010100	324	212
CNZ addr	CALL NZ, addr	C4	11000100	304	196
CP addr	CALL P, addr	F4	11110100	364	244
CPE addr	CALL PE, addr	EC	11101100	354	236
CPO addr	CALL PO,addr	E4	11100100	344	228
CZ addr	CALL Z,addr	CC	11001100	314	204
CMC	CCF	3F	00111111	077	063
CPI data	CP data	FE	11111110	376	254
CMP B	CP B	B8	10111000	270	184
CMP C	CP C	B9	10111001	271	185
CMP D	CP D	BA	10111010	272	186
CMP E	CP E	BB	10111011	273	187
CMP H	CP H	BC	10111100	274	188
CMP L	CP L	BD	10111101	275	189
CMP M	CP (HL)	BE	10111110	276	190
CMP A	CP A	BF	10111111	277	191
CMA	CPL	2F	00101111	057	047
DAA	DAA	27	00100111	047	039
DCR B	DEC B	05	00000101	005	005
DCR C	DEC C	0D	00001101	015	013
DCR D	DEC D	15	00010101	025	021
DCR E	DEC E	1D	00011101	035	029
DCR H	DEC H	25	00100101	045	037
DCR L	DEC L	2D	00101101	055	045

DCR M	DEC (HL)	35	00110101	065	053
DCR A	DEC A	3D	00111101	075	061
DCX B	DEC BC	0B	00001011	013	011
DCX D	DEC DE	1B	00011011	033	027
DCX H	DEC HL	2B	00101011	053	043
DCX SP	DEC SP	3B	00111011	073	059
DI	DI	F3	11110011	363	243
EI	EI	FB	11111011	373	251
XCHG	EX DE,HL	EB	11101011	353	235
XTHL	EX (SP),HL	E3	11100011	343	227
HLT	HALT	76	01110110	166	118
IN port	IN A, (port)	DB	11011011	333	219
INR B	INC B	04	00000100	004	004
INR C	INC C	0C	00001100	014	012
INR D	INC D	14	00010100	024	020
INR E	INC E	1C	00011100	034	028
INR H	INC H	24	00100100	044	036
INR L	INC L	2C	00101100	054	044
INR M	INC (HL)	34	00110100	064	052
INR A	INC A	3C	00111100	074	060
INX B	INC BC	03	00000011	003	003
INX D	INC DE	13	00010011	023	019
INX H	INC HL	23	00100011	043	035
INX SP	INC SP	33	00110011	063	051
JMP addr	JP addr	C3	11000011	303	195
JC addr	JP C,addr	DA	11011010	332	218
PCHL	JP (HL)	E9	11101001	351	233
JM addr	JP M, addr	FA	11111010	372	250
JNC addr	JP NC,addr	D2	11010010	322	210
JNZ addr	JP NZ, addr	C2	11000010	302	194

JP addr	JP P,addr	F2	11110010	362	242
JPE addr	JP PE,addr	EA	11101010	352	234
JPO addr	JP PO,addr	E2	11100010	342	226
JZ addr	JP Z,addr	CA	11001010	312	202
LDA addr	LD A,(addr)	3A	00111010	072	058
LDAX B	LD A,(BC)	0A	00001010	012	010
LDAX D	LD A,(DE)	1A	00011010	032	026
STA addr	LD (addr),A	32	00110010	062	050
SHLD addr	LD (addr),HL	22	00100010	042	034
STAX B	LD (BC),A	02	00000010	002	002
STAX D	LD (DE),A	12	00010010	022	018
LHLD addr	LD HL,(addr)	2A	00101010	052	042
MOV A,B	LD A,B	78	01111000	170	120
MOV A,C	LD A,C	79	01111001	171	121
MOV A,D	LD A,D	7A	01111010	172	122
MOV A,E	LD A,E	7B	01111011	173	123
MOV A,H	LD A,H	7C	01111100	174	124
MOV A,L	LD A,L	7D	01111101	175	125
MOV A,M	LD A,(HL)	7E	01111110	176	126
MOV A,A	LD A,A	7F	01111111	177	127
MOV B,B	LD B,B	40	01000000	100	064
MOV B,C	LD B,C	41	01000001	101	065
MOV B,D	LD B,D	42	01000010	102	066
MOV B,E	LD B,E	43	01000011	103	067
MOV B,H	LD B,H	44	01000100	104	068
MOV B,L	LD B,L	45	01000101	105	069
MOV B,M	LD B,(HL)	46	01000110	106	070
MOV B,A	LD B,A	47	01000111	107	071
MOV C,B	LD C,B	48	01001000	110	072
MOV C,C	LD C,C	49	01001001	111	073
MOV C,D	LD C,D	4A	01001010	112	074
MOV C,E	LD C,E	4B	01001011	113	075

MOV C,H	LD C,H	4C	01001100	114	076
MOV C,L	LD C,L	4D	01001101	115	077
MOV C,M	LD C,(HL)	4E	01001110	116	078
MOV C,A	LD C,A	4F	01001111	117	079
MOV D,B	LD D,B	50	01010000	120	080
MOV D,C	LD D,C	51	01010001	121	081
MOV D,D	LD, D,D	52	01010010	122	082
MOV D,E	LD D,E	53	01010011	123	083
MOV D,H	LD D,H	54	01010100	124	084
MOV D,L	LD D,L	55	01010101	125	085
MOV D,M	LD D, (HL)	56	01010110	126	086
MOV D,A	LD D,A	57	01010111	127	087
MOV E,B	LD E,B	58	01011000	130	088
MOV E,C	LD E,C	59	01011001	131	089
MOV E,D	LD E,D	5A	01011010	132	090
MOV E,E	LD E,E	5B	01011011	133	091
MOV E,H	LD E,H	5C	01011100	134	092
MOV E,L	LD E,L	5D	01011101	135	093
MOV E,M	LD E,(HL)	5E	01011110	136	094
MOV E,A	LD E,A	5F	01011111	137	095
MOV H,B	LD H,B	60	01100000	140	096
MOV H,C	LD H,C	61	01100001	141	097
MOV H,D	LD H,D	62	01100010	142	098
MOV H,E	LD H,E	63	01100011	143	099
MOV H,H	LD H,H	64	01100100	144	100
MOV H,L	LD H,L	65	01100101	145	101
MOV H,M	LD H,(HL)	66	01100110	146	102
MOV H,A	LD H,A	67	01100111	147	103
MOV L,B	LD L,B	68	01101000	150	104
MOV L,C	LD L,C	69	01101001	151	105
MOV L,D	LD L,D	6A	01101010	152	106
MOV L,E	LD L,E	6B	01101011	153	107

MOV L,H	LD L,H	6C	01101100	154	108
MOV L,L	LD L,L	6D	01101101	155	109
MOV L,M	LD L,(HL)	6E	01101110	156	110
MOV L,A	LD L,A	6F	01101111	157	111
MOV M,B	LD (HL),B	70	01110000	160	112
MOV M,C	LD (HL),C	71	01110001	161	113
MOV M,D	LD (HL),d	72	01110010	162	114
MOV M,E	LD (HL),E	73	01110011	163	115
MOV M,H	LD (HL),H	74	01110100	164	116
MOV M,L	LD (HL),L	75	01110101	165	117
MOV M,A	LD (HL),A	77	01110111	167	119
MVI B,data	LD B, data	06	00000110	006	006
MVI C,data	LD C,data	0E	00001110	016	014
MVI D,data	LD D,data	16	00010110	026	022
MVI E,data	LD E,data	1E	00011110	036	030
MVI H, data	LD H,data	26	00100110	046	038
MVI L,data	LD L,data	2E	00101110	056	046
MVI M,data	LD (HL),data	36	00110110	066	054
MVI A,data	LD A,data	3E	00111110	076	062
SPHL	LD SP,HL	F9	11111001	371	249
LXI B,data16	LD BC,data16	01	00000001	001	001
LXI D,data16	LD DE,data16	11	00010001	021	017
LXI H,data16	LD HL,data16	21	00100001	041	033
LXI SP,data16	LD SP,data16	31	00110001	061	049
NOP	NOP	00	00000000	000	000
ORA A	OR A	B7	10110111	267	183
ORA B	OR B	B0	10110000	260	176
ORA C	OR C	B1	10110001	261	177
ORA D	OR D	B2	10110010	262	178
ORA E	OR E	B3	10110011	263	179

ORA H	OR H	B4	10110100	264	180
ORA L	OR L	B5	10110101	265	181
ORA M	OR (HL)	B6	10110110	266	182
ORI data	OR data	F6	11110110	366	246
OUT port	OUT (port),A	D3	11010011	323	211
POP B	POP BC	C1	11000001	301	193
POP D	POP DE	D1	11010001	321	209
POP H	POP HL	E1	11100001	341	225
POP PSW	POP AF	F1	11110001	361	241
PUSH B	PUSH BC	C5	11000101	305	197
PUSH D	PUSH DE	D5	11010101	325	213
PUSH H	PUSH HL	E5	11100101	345	229
PUSH PSW	PUSH AF	F5	11110101	365	245
RET	RET	C9	11001001	311	201
RNZ	RET NZ	C0	11000000	300	192
RZ	RET Z	C8	11001000	310	200
RNC	RET NC	D0	11010000	320	208
RC	RET C	D8	11011000	330	216
RPO	RET PO	E0	11100000	340	224
RPE	RET PE	EB	11101000	350	232
RP	RET P	F0	11110000	360	240
RM	RET M	F8	11111000	370	248
RLC	RLCA	07	00000111	007	007
RRC	RRCA	0F	00001111	017	015
RAL	RLA	17	00010111	027	023
RAR	RRA	1F	00011111	037	031
RST 0	RST 0	C7	11000111	307	199
RST 1	RST 8	CF	11001111	317	207
RST 2	RST 10H	D7	11010111	327	215
RST 3	RST 18H	DF	11011111	337	223

RST 4	RST 20H	E7	11100111	347	231
RST 5	RST 28H	EF	11101111	357	239
RST 6	RST 30H	F7	11110111	367	247
RST 7	RST 38H	FF	11111111	377	255
SBI	SBC A,data	DE	11011110	336	222
SBB B	SBC A,B	98	10011000	230	152
SBB C	SBC A,C	99	10011001	231	153
SBB D	SBC A,D	9A	10011010	232	154
SBB E	SBC A,E	9B	10011011	233	155
SBB H	SBC A,H	9C	10011100	234	156
SBB L	SBC A,L	9D	10011101	235	157
SBB M	SBC A,(HL)	9E	10011110	236	158
SBB A	SBC A,A	9F	10011111	237	159
STC	SCF	37	00110111	067	055
SUI	SUB data	D6	11010110	326	214
SUB B	SUB B	90	10010000	220	144
SUB C	SUB C	91	10010001	221	145
SUB D	SUB D	92	10010010	222	146
SUB E	SUB E	93	10010011	223	147
SUB H	SUB H	94	10010100	224	148
SUB L	SUB L	95	10010101	225	149
SUB M	SUB (HL)	96	10010110	226	150
SUB A	SUB A	97	10010111	227	151
XRI	XOR data	EE	11101110	356	238
XRA B	XOR B	A8	10101000	250	168
XRA C	XOR C	A9	10101001	251	169
XRA D	XOR D	AA	10101010	252	170
XRA E	XOR E	AB	10101011	253	171
XRA H	XOR H	AC	10101100	254	172
XRA L	XOR L	AD	10101101	255	173
XRA M	XOR (HL)	AE	10101110	256	174
XRA A	XOR A	AF	10101111	257	175

Literatura

[1] An Introduction to CP/M Features and Facilities.
Digital Research, Pacifik Grove, Kalifornie 1978.

[2] Allswang, J.M.: Macintosh Definitive User's Guide.
Brady Communications Comp., Prentice Hall, Bowie, 1985.

[3] Brodský, J. – Skočovský, L.: Operační systém UNIX a jazyk C.
Praha, KVT SNTL, 1989.

[4] Clarke, A. – Eaton, J. M. – Powys–Lybbe,D.: CP/M The Software Bus ... a programmer's companion.
Sigma Technical Press, Wilmslow, Cheshire, UK 1983.

[5] CP/M Assembler (ASM) User's Guide.
Digital Research, Pacific Grove, Kalifornie 1978.

[6] CP/M Dynamic Debugging Tool (DDT) User's Guide.
Digital Research, Pacific Grove, Kalifornie 1978.

[7] CP/M 2 User' s Guide.
Digital Research, Pacifik Grove, Kalifornie 1979.

[8] Dědina, B. – Valášek, P.: Mikroprocesory a mikropočítače.
Praha, KVT SNTL, 1981.

[9] Dwyer, T. A. – Critchtfield, M.: CP/M and the Personal Computer.
Addison–Wesley, Microcomputer Books, 1983.

[10] ED: A Context Editor for The CP/M Disk System User' s Manual.
Digital Research, Pacifik Grove, Kalifornie 1978.

[11] Granát, L. – Sechovský,H.: Počítačová grafika.
Praha, KVT SNTL, 1980.

[12] Hansen, P. B.: Principy operačních systémů.
Praha, KVT SNTL, 1983.

[13] Hogan, T.: Osborne CP/M User Guide (druhé vydání).
Osborne/McGraw-Hill, Berkeley, Kalifornie 1982.
[14] Johnson-Laird, A.: The Programmer's CP/M Handbook.
Osborne/McGraw-Hill, Berkeley, Kalifornie 1983.
[15] Kernighan, B. W. - Pike, R.: The UNIX Programming Environment.
Prentice-Hall, New Jersey 1984.
[16] Kildall, G.: CP/M: A Family of 8- and 16-Bit Operating Systems.
BYTE, s. 216-231, June 1981.
[17] MACRO-80 Assembler Reference Manual, Microsoft, 1980.
[18] Madnick, S. E. - Donovan,J.J.: Operační systémy.
Praha, KVT SNTL, 1981.
[19] Norton, P.: Inside the IBM PC. Prentice-Hall 1983.
[20] Norton, P.: Programmer' s Guide to The IBM PC.
Washington, Microsoft Press, 1985.
[21] Plášil, F.: Operační systémy. Skripta ČVUT FEL, Praha 1980.
[22] Plášil, F. - Richta, K. - Tschernoster,E. - Zajíc,J.: Operační systémy
osobních počítačů. In: SOFSEM '86, díl II., s. 1-38,
ÚVT UJEP Brno + JČSMF, Liptovský Ján 1986.
[23] Plášil, F. - Staudek,J.: Operační systémy.
Připravováno k publikaci v SNTL.
[24] Ribarič, S.: Architektura mikroprocesora. Tehnička knjiga, Zagreb 1985
Překlad: Architektury mikroprocesorů. Bratislava, ALFA, 1989.
[25] Richta, K. - Zajíc, J.: Operační systém typu CP/M pro mikropočítače.
Knižnice ČSVTS Mikroprocesorová technika, svazek 14, díl 1,
Praha 1986.
[26] Slípka, J.: Navrhování mikroprocesorových systémů.
Praha, SNTL, 1985.
[27] Starý, J.: Mikropočítač a jeho programování.
Praha, KVT SNTL, 1987.
[28] Tschernoster, E. - Vlachovský, V.: Operační systém MS-DOS.
Připravováno k publikaci v SNTL.