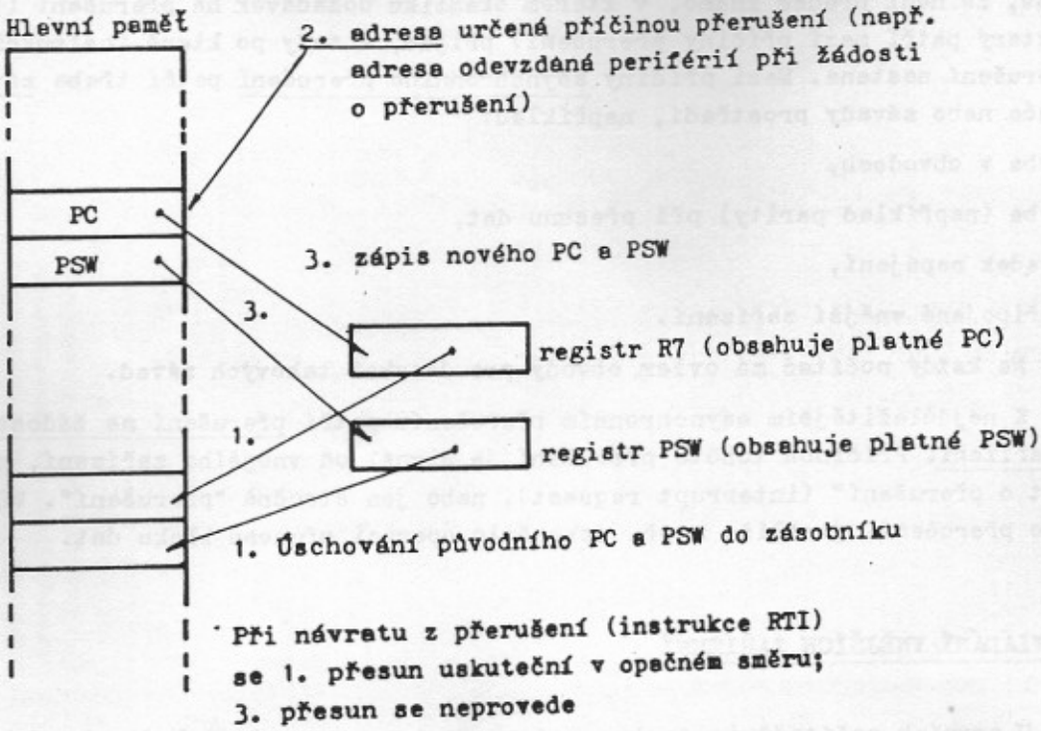



Obr. 5.4.1 Činnost při přerušení na počítači SM

vat výjimky (exceptions). Oproti programovanému přerušení je odlišnost v tom, že z provedení příslušné instrukce (třeba sčítání) v programu plyne pouze možnost přerušení. K určení, zda k přerušení skutečně dojde, je potřebné znát hodnoty operandů v okamžiku provádění instrukce.

Výjimek bývá ve strojovém jazyce velmi mnoho. Typické jsou následující:

1. přeplnění v pevné řádové čárce,
2. dělení nulou,
3. přeplnění v pohyblivé řádové čárce,
4. nenaplnění v pohyblivé řádové čárce,
5. nedovolená adresa v instrukci (např. lichá adresa při skoku),
6. adresa mimo rozsah hlavní paměti,
7. neznámý operační znak,
8. privilegovaná instrukce při činnosti v režimu "uživatel",
9. nedovolený tvar operandu,
10. nedovolené určení operandu (např. nesprávné číslo registru),
11. přeplnění zóny pro zásobník,
12. výběr z prázdného zásobníku,
13. nedovolený přístup do paměti (porušení ochrany paměti),
14. zápis nedovoleného údaje do některého registru procesoru (např. do registru stavového slova procesoru),
15. nedovolený údaj při překladu adres.

Programovaná přerušení a přerušení způsobená výjimkami bývají souhrnně nazývána synchronní přerušení. Nastávají totiž bezprostředně po instrukci, která je způsobila; tedy synchronně vzhledem k průběhu programu.

Ostatní druhy přerušení jsou asynchronní vzhledem k průběhu programu. To znamená, že není předem známo, v kterém okamžiku požadavek na přerušení (tj. signál, který patří mezi příčiny přerušení) přijde, a tedy po které instrukci programu přerušení nastane. Mezi příčiny asynchronního přerušení patří třeba závady počítače nebo závady prostředí, například:

a) chyba v obvodech,

b) chyba (například parity) při přesunu dat,

c) výpadek napájení,

d) nepřipojené vnější zařízení.

Ne každý počítač má ovšem obvody pro detekci takových závad.

K nejdůležitějším asynchronním přerušením patří přerušení na žádost vnějšího zařízení. Příčinou tohoto přerušení je signál od vnějšího zařízení, nazývaný "žádost o přerušení" (interrupt request), nebo jen stručně "přerušení". VZ může žádat o přerušení, jestliže např. dokončilo operaci přenosu bloku dat.

## 4.5 OVLÁDÁNÍ VNĚJŠÍCH ZAŘÍZENÍ

4.5.1 U prvních počítačů instrukce vstupu a výstupu trvala tak dlouho jako odpovídající činnost vnějšího zařízení. Například provedení k-instrukce "přečti znak z děrné pásky do střadače" sestávalo z následujících kroků:

a) test připravenosti snímače děrné pásky,

b) generování signálů určujících požadovanou operaci,

c) čekání na signál o dokončení operace čtení,

d) zápis přečteného znaku do střadače.

Při tomto postupu buď pracovala základní jednotka a nepracovalo vnější zařízení, nebo naopak pracovalo vnější zařízení a nepracovala základní jednotka (bod c). Nevýhoda byla v tom, že většina vnějších zařízení je oproti procesoru pomalá, a tedy procesor by byl velmi špatně využit.

4.5.2 Lepší řešení je instrukcí vstupu a výstupu předávat vnějšímu zařízení pouze příkaz k provedení akce. Tím instrukce vstupu a výstupu skončí a provádí se následující instrukce k-programu. V tomto okamžiku pracuje procesor i vnější zařízení současně; výsledky požadované operace vstupu a výstupu však nejsou ještě k dispozici. Výsledky lze použít až po ukončení příslušné operace vnějšího zařízení. Užívá se několik způsobů, jak takovou událost zjistit v procesoru:

a. Je známa doba trvání operace vnějšího zařízení, tj. po uplynutí tohoto času je k dispozici výsledek (či příznak chyby). Známe-li dobu provádění instrukcí procesoru, můžeme spočítat, od kterého místa v programu po instrukci vstupu a výstupu je výsledek operace vnějšího zařízení použitelný. Tento způsob se užívá poměrně zřídka a pouze u velmi jednoduchých počítačů a vnějších zařízení, jako jsou například čidla a převodníky. Některá tato zařízení pracují dokonce bez příkazu k provedení operace. Provádějí jednou operaci (ale zato pořád) a údaje jsou k dispozici v pravidelných intervalech.

U většiny počítačů je však výpočet doby trvání instrukce nepříjemná úloha. Někdy však postačuje použít hrubý odhad nebo lze ponechat značnou časovou rezervu pro dokončení činnosti vnějšího zařízení.

B. Vnější zařízení oznamuje ukončení operace. To znamená, že je nastaven (např. v určeném registru) nějaký bit (tzv. příznak ukončení operace), který lze testovat programem. Testování lze řešit podle následujícího schématu:

1. Instrukce vstupu a výstupu,

2. }
. } jakékoli instrukce nepoužívající výsledek
. } operace vnějšího zařízení,
n. }

n + 1. podmíněný skok na "n + 1", jestliže příznak ukončení není nastaven,

n + 2. další instrukce (výsledek operace vnějšího zařízení je k dispozici).

.
.
.

Opakování instrukce n + 1 je nazýváno činné čekání. Procesor sice pracuje, ale "nedělá nic užitečného". Činné čekání nastane v případě, že operace vnějšího zařízení neskončí před provedením instrukce n + 1; na instrukci n + 2 se v každém případě přejde až po skončení operace vnějšího zařízení. Činné čekání připomíná bod "c" odstavce 4.5.1. Zlepšení je v tom, že byly provedeny "užitečné" instrukce 2 až n.

Může však nastat i situace, že vnější zařízení operaci ukončilo před instrukcí n + 1 a program tuto skutečnost "vezme na vědomí" až instrukcí n + 1. Vnější zařízení tedy určitou dobu nepracuje, i když další požadavky na jeho práci třeba jsou. Vyhnout se oběma nevýhodám (špatné použití procesoru či vnějšího zařízení) však znamená řešit otázku doby provádění instrukcí a doby trvání operací vnějších zařízení. To je ovšem pro některá vnější zařízení neřešitelné s potřebnou přesností a pro jiná může být takový požadavek i nesmyslný.

C. Vnější zařízení oznamuje ukončení operace signálem a tento signál je v procesoru využíván jako požadavek na přerušení (odst. 4.4.6). Přerušení má za následek provedení obslužného programu, v kterém se může ihned provést další potřebná instrukce vstupu či výstupu. Tím je umožněno, aby vnější zařízení pracovalo maximální rychlostí, dokud je pro něj připravena "zásoba úkolů". To je u některých vnějších zařízení (např. u diskových pamětí) obvykle velmi důležité pro celkovou výkonnost počítačového systému.

Oproti způsobu B je též zřejmé, že test ukončení operace vnějšího zařízení by měl být umístěn až těsně před první instrukcí používající výsledek operace vnějšího zařízení.

## 4.6 PROSTŘEDKY POČÍTAČE

4.6.1 Má-li na jednom k-počítači být implementován jeden nebo několik o-počítačů, je třeba specifikovat procesor, paměť a vnější zařízení těchto o-počítačů

a řešit otázku jejich implementace pomocí prostředků k-počítače. Prostředky k-počítače (stručně k-prostředky) rozumíme procesor, hlavní paměť a vnější zařízení. Tyto prostředky byly charakterizovány operacemi, které mohou provádět. Souhrnně vzato jde o následující operace:

| Prostředek | Operace |
|---|---|
| Procesor (k-procesor) | Instrukce<br>Přerušení<br>Spuštění |
| Hlavní paměť (k-paměť) | Čtení z určené adresy<br>Zápis na určenou adresu |
| Vnější zařízení (k-periférie) | Příkazy určené operacemi vstupu nebo výstupu |

Výsledky operací ovšem obecně vzato závisí na předchozích operacích. Vliv těchto předchozích operací na výsledek dané operace může být vyjádřen stavem prostředku. Například operace čtení hlavní paměti závisí na posledním zápise na stejnou adresu. Stav hlavní paměti je pak dán obsahem všech paměťových míst. Obdobně stav k-procesoru bývá dán obsahem registru stavového slova procesoru a obsahem dalších registrů přístupných k-programu.

4.6.2 Obdobně jako jsou specifikovány k-prostředky, můžeme specifikovat prostředky jednotlivých obslužných počítačů, tedy prostředky úrovně 3. Pro rozlišení budeme tyto prostředky nazývat obslužné prostředky nebo o-prostředky. o-prostředky můžeme charakterizovat podobně jako k-prostředky:

| Prostředek | Operace |
|---|---|
| Procesor | Příkazy výkonného jazyka<br>Přerušení výkonného jazyka<br>Příkazy řídícího jazyka<br>Spuštění o-počítače |
| Hlavní paměť (o-paměť) | Čtení z určené adresy<br>Zápis na určenou adresu |
| Vnější zařízení (o-periférie) | Příkazy určené operacemi vstupu a výstupu o-jazyka |

Zatím jsme se zmínili jen o některých z uvedených operací, podrobnější vysvětlení postupně následuje.

4.6.3 O prostředcích o-počítačů se v publikacích o operačních systémech tradičně hovoří jako o virtuálních prostředcích. Pro odlišení se pak k-prostředky nazývají reálné prostředky. Taková terminologie vyhovuje, jestliže se hovoří jen o jedné vrstvě (v tomto případě o operačním systému).

V této učebnici se však probírá více vrstev a při použití uvedené terminologie bychom se těžko vyhýbali nedorozuměním. Např. pokud bychom hovořili o vrstvě

mikroprogramů, jevily by se nám prostředky úrovně 1 (například procesor mikroinstrukcí a řídící paměť) jako reálné a k-prostředky jako virtuální.

4.6.4 Operační systém implementuje o-prostředky a používá k tomuto účelu prostředky, jimiž disponuje, tedy k-prostředky.

Například k-procesor provádí své operace, což se na úrovni 3 jeví jako činnost o-procesoru. V tomto případě říkáme, že k-procesor je použit (přidělen) k implementaci o-procesoru nebo že k-procesor je přidělen (k interpretaci) o-programu.

Složitější situace vzniká, má-li být implementováno více o-procesorů pomocí jednoho k-procesoru, například při vytváření multiprogramního operačního systému na jednoprocesorovém počítači. V tomto případě k-procesor střídavě interpretuje jednotlivé o-programy; této situaci říkáme, že k-procesor je sdílen jednotlivými o-procesory. Sdílení je řešeno odnímáním a vracením (opětovným přidělováním) k-procesoru jednotlivým o-procesorům. Odejmutí a vrácení k-procesoru libovolnému o-procesoru ovšem nemá mít vliv na výsledek provádění o-programu (má ovšem vliv na rychlost jeho provádění). Tuto podmínku obecněji vyjadřujeme jako požadavek zachování stavu o-prostředku (odst. 4.6.1) při odejmutí a vrácení k-prostředku.

U některých o-prostředků však může zachování stavu při sdílení působit obtíže. Proto je odnímání odpovídajících k-prostředků povoleno jen ve zvláštních situacích, signalizovaných o-programem. Mezním případem je pak přidělení k-prostředku o-programu na jeho začátku a odejmutí po jeho ukončení.

Z uvedeného hlediska můžeme k-prostředky rozdělit na odnímatelné a na pevně přidělované. Pevně přidělený prostředek tedy lze odnímat jen za předpokladu spolupráce o-programu, neboť jinak je obtížné zachovat stav příslušného o-prostředku.

4.6.5 Některé k-prostředky nejsou přidělovány jako celek, ale jsou přidělovány po částech. Příkladem takového prostředku je hlavní paměť nebo disková paměť. Je-li o-programu přidělena jen část k-prostředku, je třeba zajistit, aby zbývající části nemohl používat (ať již úmyslně, nebo v důsledku chyby). Takovou kontrolu provádí operační systém, avšak zpravidla ji nemůže provádět kompletně, neboť interpretace výkonného jazyka je jen částečná. Neprivilegované instrukce patřící do o-programu totiž provádí přímo k-počítač. Má-li být tedy hlavní paměť přidělována počástech, musí k-počítač "umět hlídat" přístup do jednotlivých částí paměti.

4.6.6 V některých případech je způsob implementace o-prostředku značně složitý. Například o-paměť může být implementována pomocí hlavní paměti a diskové paměti, přičemž k-počítač překládá (transformuje) adresy všech instrukcí. V tomto případě jsou o-programu pevně přiděleny určité části diskové paměti a části hlavní paměti jsou mu přidělovány a odnímány dle potřeby. Touto problematikou se zabýváme v podkapitole 4.8.

## 4.7 VNĚJŠÍ ZAŘÍZENÍ O-POČÍTAČE

4.7.1 Vnější zařízení o-počítače jsme v odst. 5.6.2 stručně nazvali o-periférie. o-periférie může být implementována pomocí pevně přidělené k-periférie a obslužných programů pro tuto k-periférii.

Například o-program provádí operace tisku, tedy pracuje s o-periférií tiskárna. S tímto vnějším zařízením úrovně 3 se však provádí jiné operace než provádí k-program s vnějším zařízením tiskárna. Jistěže fyzicky vzato jde v obou případech o provedení požadovaného tisku. Programátora nezajímá "tiskárna jako fyzický objekt", ale možné způsoby jejího ovládání, potřebné k dosažení tisku. Z hlediska programátora tedy není o-periférie tiskárna totéž co k-periférie tiskárna. Potřebujeme-li tyto dva abstraktní pohledy na tiskárnu rozlišit, používáme označení o-tiskárna a k-tiskárna (obdobně pro jiné vnější zařízení). Na proces "ovládání o-tiskárny" se lze dívat jako na abstrakci procesu "ovládání k-tiskárny".

4.7.2 o-periférie je specifikována operacemi (odst. 4.6.1 a 4.6.2), tedy nezávisle na způsobu implementace. To umožňuje řešit implementaci o-periférie různými způsoby.

Například o-tiskárna může být implementována tak, že údaje určené k tisku jsou postupně nejdříve uloženy na vnější paměť a po skončení o-programu teprve operační systém dává příkazy této k-periférii (tj. fyzicky k tisku dojde až po skončení o-programu).

Rozlišujeme tedy nejen o-periférie a k-periférie, ale též různé způsoby implementace o-periférie s využitím jedné nebo více k-periférií.

U mnohých o-počítačů však počet o-periférií nesouhlasí s počtem k-periférií stejného druhu. K tomu vedou např. následující důvody:

1. určité k-periférie mohou být vyhrazeny pouze pro činnost operačního systému a na úrovni 3 jsou tedy "neviditelné" (např. jedna magnetická páska je vyhrazena operačnímu systému);
2. multiprogramní operační systém může jednotlivé k-periférie přidělit různým o-programům;
3. operační systém může implementovat více o-periférií, než je k-periférií.

Pro vysvětlení posledního bodu připomeňme způsob implementace o-tiskárny z odstavce 4.7.2. Bude-li implementace provedena takovým způsobem, může mít o-počítač více tiskáren a vždy po skončení oáprogramu jsou pak postupně soubory údajů pro jednotlivé o-tiskárny vytisknuty na k-tiskárně.

4.7.3 V odstavci P 4.8 se zabýváme instrukcemi vstupu a výstupu, pomocí kterých lze řídit operace k-zařízení. Operace čtení a zápisu pracují s daty (tj. data jsou jejich parametry), která je třeba předávat mezi k-procesorem a k-zařízením. V některých případech jsou předávány jednotlivé znaky, u jiných zařízení jsou předávány bloky bytů (či slabik). Pro sjednocení můžeme položky dat přenášené mezi procesorem a vnějším zařízením nazvat záznam (record). Tyto záznamy jsou většinou na médiu vnějšího zařízení ukládány jako celek (např. blok na magnetické pásce, záznam v sektoru disku, řádek na tiskárně).

Operace čtení a zápisu pro o-zařízení pracují rovněž s daty, obvykle však odlišné délky. Položky dat přenášené mezi o-procesorem a o-periférií nazveme tedy o-záznam a položky dat přenášené mezi k-procesorem a k-periférií nazveme k-záznam.

P o z n á m k a : o-záznam je často nazýván "logický záznam" nebo "věta". k-záznam je též nazýván "fyzický záznam" nebo "blok".

Implementace o-periférie pomocí k-periférie tedy určuje i vztah mezi o-záznamy a k-záznamy. Typické situace jsou následující (v běžnější terminologii):

- věta stejně dlouhá jako blok,
- několik vět je v jednom bloku,
- věta nemusí být celá v jednom bloku.

Další možnosti jsou užívání proměnné délky k-záznamu a (častěji) užívání proměnné délky o-záznamu.

4.7.4 Nosiče záznamů (média) jsou rozdělovány na určité fyzické celky, nazývané většinou kotouč nebo svazek (reel, volume, deck, pack). Operační systém napomáhá při identifikaci a vedení evidence těchto svazků. K tomuto účelu obvykle slouží první k-záznam na kotouči, nazývaný záhlaví nebo počáteční návěští kotouče (volume header, volume label). Kotouč, resp. svazek může obsahovat určitý počet k-záznamů (podle velikosti kotouče a délky záznamů). Používáme-li určité vnější zařízení, je třeba se seznámit:

- s možnými délkami bloků,
- s kapacitou kotoučů, resp. svazků,
- s využitelností kotoučů, resp. svazků (např. kolik bloků jaké délky lze zaznamenat).

Opět z hlediska problémů řešených na počítači jsou velikosti kotoučů, resp. svazků zpravidla nezajímavé; taková omezení program jen komplikují. Na úrovni 3 proto pracujeme s logickými celky nazývanými soubor. Sktriktně vzato soubor je evidovaná posloupnost vět, zapsaná mezi "otevřením" a "uzavřením" souboru. Evidování se zpravidla provádí přiřazením jména souboru a dalších údajů, jako je datum vzniku, číslo verze apod. Tyto údaje jsou uvedeny v počátečním návěští souboru a případně v koncovém návěstí souboru (což jsou k-záznamy standardní délky a tvaru).

Implementace o-periférie pomocí k-periférie tedy určuje i vztah mezi soubory a kotouči, resp. svazky. Některé operační systémy např. podstatně napomáhají použití

- vícesouborových kotoučů magnetických pásek,
- vícekotoučových souborů na magnetických páskách,
- vícesouborových svazků diskových pamětí,
- vícesvazkových souborů na diskových pamětech.

Pokud se nejedná o výměnná média, resp. pokud uživatel nemůže manipulovat s jednotlivými médii, nemusí uživatel rozmístění souborů na médiích vůbec znát.

4.7.5 Při definování operací o-periférií se projevuje též snaha o odstranění nepodstatných odlišností mezi nimi. o-periférie některých druhů mají tedy způsob ovládání shodný. Podstatné odlišnosti jsou ve způsobu přístupu k údajům, který odpovídající k-periférie umožňuje (postupný přístup nebo přímý přístup).

Při vzniku souboru jsou jeho věty zapisovány postupně. Je-li toto pořadí zachováno, hovoříme o postupně (sekvenčně) uspořádaném souboru. Není-li tomu tak, lze věty zpravidla nalézt podle jejich adresy nebo podle jejich zvláštní části (klíče), určené k identifikaci věty.

Operace definované pro o-periférii tedy umožňují postupný přístup nebo přímý přístup nebo oba přístupy. V případě postupného přístupu musí být soubor sekvenční, v případě přímého přístupu soubor může či nemusí být sekvenčně uspořádaný.

o-periférie s postupným přístupem zpravidla může provádět zápis (tiskárna), čtení (snímač děrných štítků), případně zápis i čtení (magnetická páska nebo psací stroj).

Operace zápisu a čtení mohou být vzájemně závislé různým způsobem. Např. magnetická páska je paměťové zařízení s postupným přístupem, tedy zapsané věty lze v daném pořadí znovu přečíst. Psací stroj takovou vlastnost ovšem nemá.

4.7.6 Jak již bylo uvedeno, příkazy pro o-periférie mají formu programovaných přerušení. Parametry těchto příkazů jsou většinou ukládány do registrů.

Pro o-periférie s postupným přístupem jsou typicky používány následující příkazy (TPS je zkratka pro tabulku popisu souboru):

Tabulka 5.7.1

Tabulka popisu souboru

| Příkaz | Parametry | Význam (podrobnosti viz dále) |
|---|---|---|
| OPEN | adresa TPS | otevření souboru |
| GET | adresa TPS<br>adresa oblasti HP | čtení (kopírování věty souboru do určené oblasti HP) |
| PUT | adresa TPS<br>adresa věty v HP | zápis (kopírování věty z HP do souboru) |
| CLOSE | adresa TPS | uzavření souboru |

Tabulka popisu souboru je datová struktura o-programu, která určuje například:

- jméno a evidenční údaje v souboru,
- druh souboru (vstupní, výstupní),
- druh o-periférie,
- délku věty,
- délku bloku,
- adresu pro výjimku "konec čtení",
- adresu pro výjimku "chyba".

Příkaz OPEN určuje, že operační systém má připravit možnost užívání o-periférie. Může být tedy využit jako "žádost o přidělení k-periférie". Jde-li o vstupní soubor, lze též přečíst údaje z k-periférie do vyrovnávacích pamětí. Někdy je druh souboru konkretizován až příkazem OPEN (další parametr).

Příkaz GET lze užít jen pro vstupní soubor. Věta, která je na řadě, je přesunuta z vyrovnávacích pamětí do oblasti HP určené tímto příkazem. Je-li třeba, je též spuštěna další operace, čtení z periférie do vyrovnávací paměti.

Příkaz PUT lze užít jen pro výstupní soubor. Věta určená tímto příkazem je přesunuta z oblasti HP do vyrovnávací paměti. Je-li třeba, je též spuštěna další operace zápisu z vyrovnávací paměti na periférii.

Příkaz CLOSE určuje, že operační systém má ukončit práci s o-periférií. Může být tedy využit jako "souhlas k odnětí k-periférie". Jde-li o výstupní soubor, je třeba údaje z vyrovnávacích pamětí ještě zapsat na periférii. Někdy je dalším parametrem příkazu ještě možno určit manipulaci se souborem (ponechání, převinutí, uvolnění).

Předchozí výklad připouští možnost, aby operační systém používal jednu či více vyrovnávacích pamětí pro uložení k-záznamů. Užití více vyrovnávacích pamětí může zrychlit provádění o-programu (snížením nepříznivého vlivu nerovnoměrnosti v požadavcích o-programu na činnost periférie).

Ještě je potřebné vysvětlit význam adresy pro výjimku "konec čtení" v tabulce popisu souboru. Zatím jsme předpokládali, že příkazy o-programu jsou prováděny bez výjimek obsluhovaných o-programem. Pokud je však čten soubor, jehož počet vět nelze při psaní o-programu určit, je třeba testovat konec souboru. Jedna možnost by byla definovat příkaz "skok, je-li vstupní soubor na konci" a tento test provádět před každým příkazem GET. Jednodušší řešení však je stanovit, že provede-li o-program operaci GET a v čteném souboru již není věta, nastane výjimka o-programu. Tabulka popisu souboru (TPS) tedy obsahuje adresu, na které má činnost o-programu pokračovat v případě výjimky při provádění příkazu čtení tohoto souboru.

Obdobně je používána výjimka "chyba"; na určenou adresu se může skákat například v případě chyby v návěstí souboru, při nalezené nesprávně dlouhé věty apod. Bývá stanoven i způsob návratu po "obsluze" výjimky.

P o z n á m k a : Dříve jsme se seznámili s výjimkami, přerušeními a s návraty z přerušení na úrovni 2. Nyní vidíme, že obdobný mechanismus je užíván i na úrovni 3, tj. mohli bychom mluvit o o-výjimkách, o-přerušeních, o obslužných o-programech a návratu z nich, a to na rozdíl od k-výjimek atd. Uvedený způsob "řízení následnosti" se někdy užívá i v mikroprogramování a je použit i v některých novějších vyšších programovacích jazycích.

4.7.7 Pro přímý přístup do souboru je třeba operace čtení a zápisu definovat poněkud odděleně. Operaci čtení je třeba doplnit parametrem určujícím, která věta má být čtena. Operaci zápis je třeba doplnit parametrem určujícím, kam má být věta zapsána. Jak se tato určení provádějí, souvisí s organizací příslušného souboru.

4.7.8 Pevné přidělování k-periférií jednotlivým o-programům není pro multiprogramní operační systém většinou příliš vhodné. Většina o-programů zpravidla potřebuje stejné periférie (například snímač štítků a tiskárnu). o-programy pak čekají na přidělení příslušné periférie, zatímco o-program, který ji má přidělenu, ji třeba nevyužívá na plný výkon.

U operačních systémů účastnického typu je ostatně nežádoucí, aby počet účastníků, kteří mohou souběžně pracovat, byl omezen jinak než počtem terminálů (omezení třeba počtem tiskáren nebo magnetopáskových jednotek je tedy neúčelné). Přitom je třeba předpokládat, že všichni účastníci mohou požadovat stejné o-periférie.

Proto je výhodné u takových operačních systémů před spuštěním o-programu všechny vstupní soubory překopírovat na vnější paměť, potom provést o-program a na závěr výstupní soubory překopírovat z vnější paměti na příslušné k-periférie. Při provádění o-programu jsou tedy všechny soubory na vnější paměti: autor o-programu se však o to nemusí starat. Za výjimku lze považovat soubor na terminálu, neboť terminál je přidělen příslušnému účastníkovi. Další výjimkou může být rozsáhlý soubor (např. na magnetické pásce), jehož kopii je nevýhodné nebo nemožné mít na vnější paměti.

4.7.9 Poměrně složitá situace vzniká při sdílení diskových pamětí. Operační systém obvykle přiděluje o-programu části diskové paměti, což se z hlediska o-programu jeví jako možnost práce s o-periférií disková paměť (resp. zóna diskové paměti). Určité části diskových pamětí jsou využívány pro uložení operačního systému, dále pro uložení o-programů připravených ke spuštění, pro uložení kopírovaných vstupních a výstupních souborů apod. Je-li diskových jednotek více, lze některé přidělit o-programu jako celek.

## 4.8 HLAVNÍ PAMĚŤ O-POČÍTAČE

4.8.1 k-program pracuje s údaji uloženými v hlavní paměti (k-paměti). Obdobně o-program pracuje s údaji uloženými v hlavní paměti o-počítače (o-paměti). o-paměť se od k-paměti odlišuje mimo jiné proto, že část k-paměti je třeba pevně přidělit operačnímu systému. Dalším důvodem je snaha, aby používání počítače bylo na úrovni 3 jednodušší než na úrovni 2, což se vztahuje i na používání hlavní paměti.

o-paměť má zpravidla stejně velká paměťová místa jako k-paměť a při práci s ní jsou užívány stejné operace - zápis a čtení. Rozdíl je v počtu paměťových míst a někdy též v nestálém přiřazení paměťových míst o-paměti paměťovým místům k-paměti (o-program může být i v k-paměti přemisťován). Záleží tedy na tom, jak jsou operace zápisu a čtení o-paměti pomocí operací zápisu a čtení k-paměti implementovány.

4.8.2 U monoprogramního operačního systému je část k-paměti přidělena operačnímu systému a zbytek k-paměti je použit "jako" o-paměť, tj. pro implementaci o-paměti. Implementace ovšem je v tomto případě jednoduchá, postačí adresám o-pa-

měti (o-adresám) přiřadit adresy k-paměti (k-adresy). To lze provést například tak, že po vyhodnocení o-adresy je k ní přičtena adresa začátku výkonného o-programu v k-paměti (adresování s bázovým registrem - odst. 2.4.7). Překročení rozsahu o-paměti lze pak zjistit podle překročení rozsahu k-paměti.

P ř í k l a d

Uvedené řešení je užíváno na počítačích Tesla 200/300, kde k o-adrese je přičítán obsah zvláštního registru, registru ROP (registr počátku programu). Obsah registru může měnit jen supervizor.

Uvedený postup je elementárním případem tzv. dynamického překladu adres. Přičítání obsahu uvedeného zvláštního registru se totiž provádí před každým použitím o-adresy znovu.

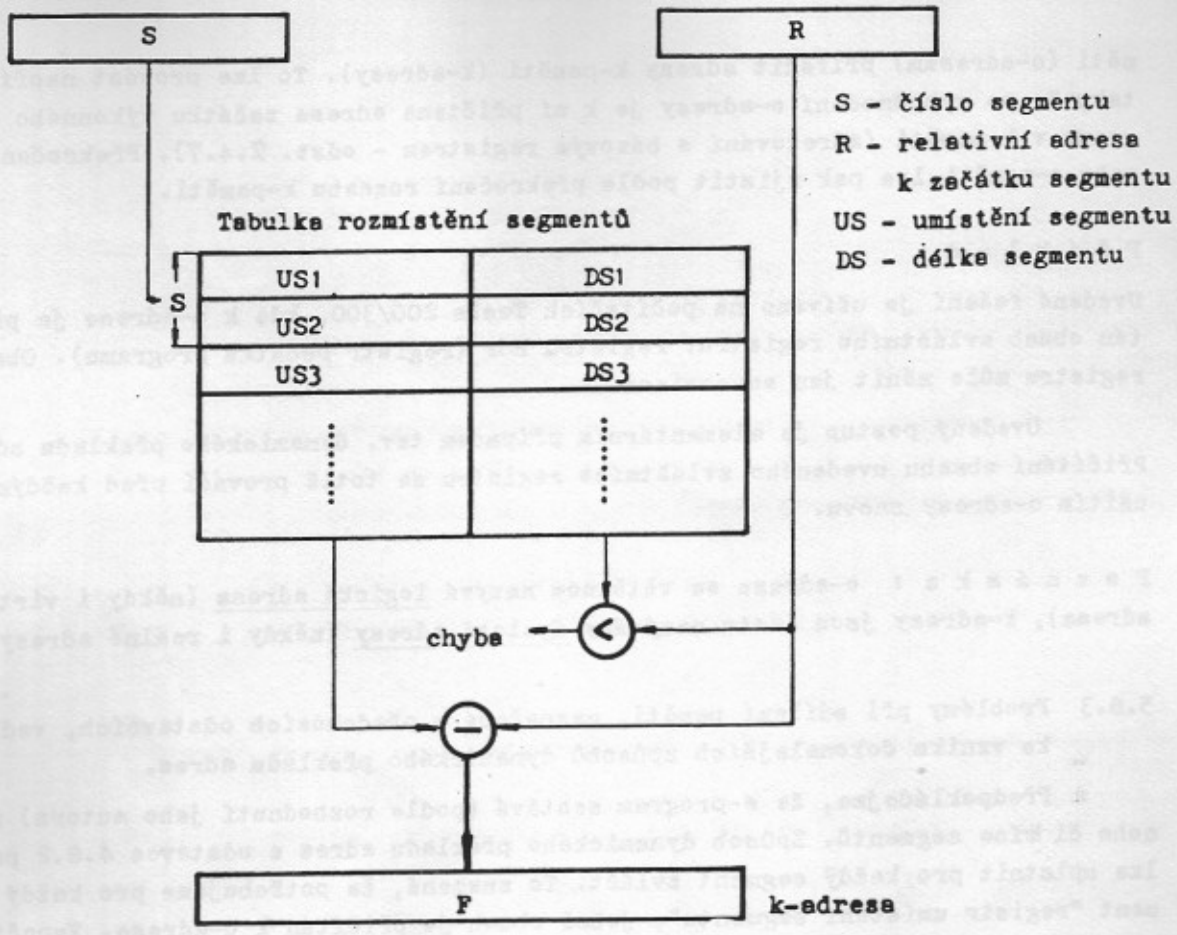
P o z n á m k a : o-adresa se většinou nazývá logická adresa (někdy i virtuální adresa), k-adresy jsou často nazývány fyzické adresy (někdy i reálné adresy).

5.8.3 Problémy při sdílení paměti, naznačené v předchozích odstavcích, vedly ke vzniku dokonalejších způsobů dynamického překladu adres.

Předpokládejme, že o-program sestává (podle rozhodnutí jeho autora) z jednoho či více segmentů. Způsob dynamického překladu adres z odstavce 4.8.2 pak lze uplatnit pro každý segment zvlášť. To znamená, že potřebujeme pro každý segment "registr umístění segmentu", jehož obsah je přičítán k o-adrese. Vypočtenou k-adresu je třeba kontrolovat, zda nepřekračuje stanovenou délku segmentu. Pro každý segment tedy potřebujeme ještě "registr délky segmentu". Obsahy těchto registrů dohromady tvoří tabulku umístění segmentů (u některých počítačů je tato tabulka v hlavní paměti a ne ve zvláštních registrech).
o-adresa sestává tedy z čísla segmentu (S) a z adresy relativní k začátku segmentu (R). Čísla segmentu je použito pro adresování tabulky umístění segmentů.
o-adresy lze rozdělit na adresy ukazující dovnitř segmentu a mimo segment.
Vnitřní o-adresy mohou sestávat jen z položky "R", neboť "S" lze užít implicitně.

Uvedený způsob může být zdokonalen tak, aby při zpracování o-programu nemusely být všechny segmenty umístěny v k-paměti. Tabulku rozmístění segmentů např. doplníme bitem "přítomnost v hlavní paměti". Jestliže se překládá o-adresa, provede se nejdříve test přítomnosti. Je-li segment nepřítomen, nelze o-adresu přeložit, a tedy nelze provést instrukci, v níž je adresa použita. Proto se namísto překladu adresy provede přerušení, přičemž návratová adresa ukazuje na neprovedenou instrukci. Příčinou tohoto přerušení je tedy chybějící segment. Obslužný program pro toto přerušení překopíruje potřebný segment z vnější paměti do k-paměti namísto jednoho či více jiných segmentů, které případně překopíruje zpět na vnější paměť a v tabulce rozmístění segmentů označí jako nepřítomné.
V případě potřeby lze segmenty i přemístit v hlavní paměti, postačí upravit záznamy v tabulce rozmístění segmentů.

Přesouvání segmentů mezi vnější pamětí a hlavní pamětí má za následek, že program někdy musí čekat na přístup k segmentům. Při multiprogramování to však nebývá na závadu, neboť lze zatím pracovat na jiném programu. Jestliže segmenty




Obr. 5.8.1 Překlad adres při segmentaci

číslujeme v rámci programu, potřebujeme pro každý program jinou tabulku rozmístění segmentů (což lze řešit například změnou obsahu registrů umístění a délky segmentu).

Uvedený způsob je nazýván segmentace. Umožňuje například:

- přidělit segmentům oblast k-paměti odpovídající délce segmentu,
- jednoduché přemísťování segmentů,
- umístit v k-paměti jen část programu (program může být i delší než hlavní paměť, vyměňování je poměrně rychlé),
- řešit sdílení paměti při multiprogramování (včetně využití větší hlavní paměti, než je jedna o-paměť),
- řešit sdílení segmentů,
- považovat soubory umístěné na vnější paměti za datové segmenty programu.

Nevýhodou je zejména rozdílná délka segmentů, která vede k horšímu využití hlavní paměti nebo ke ztrátám času při přemísťování segmentů uvnitř k-paměti.

P o z n á m k a : V současné době se často setkáváme s počítači, které byly původně vyvinuty bez dynamického překladu adres (PDP 11/20, SM-3, IBM 360, JSEP I)

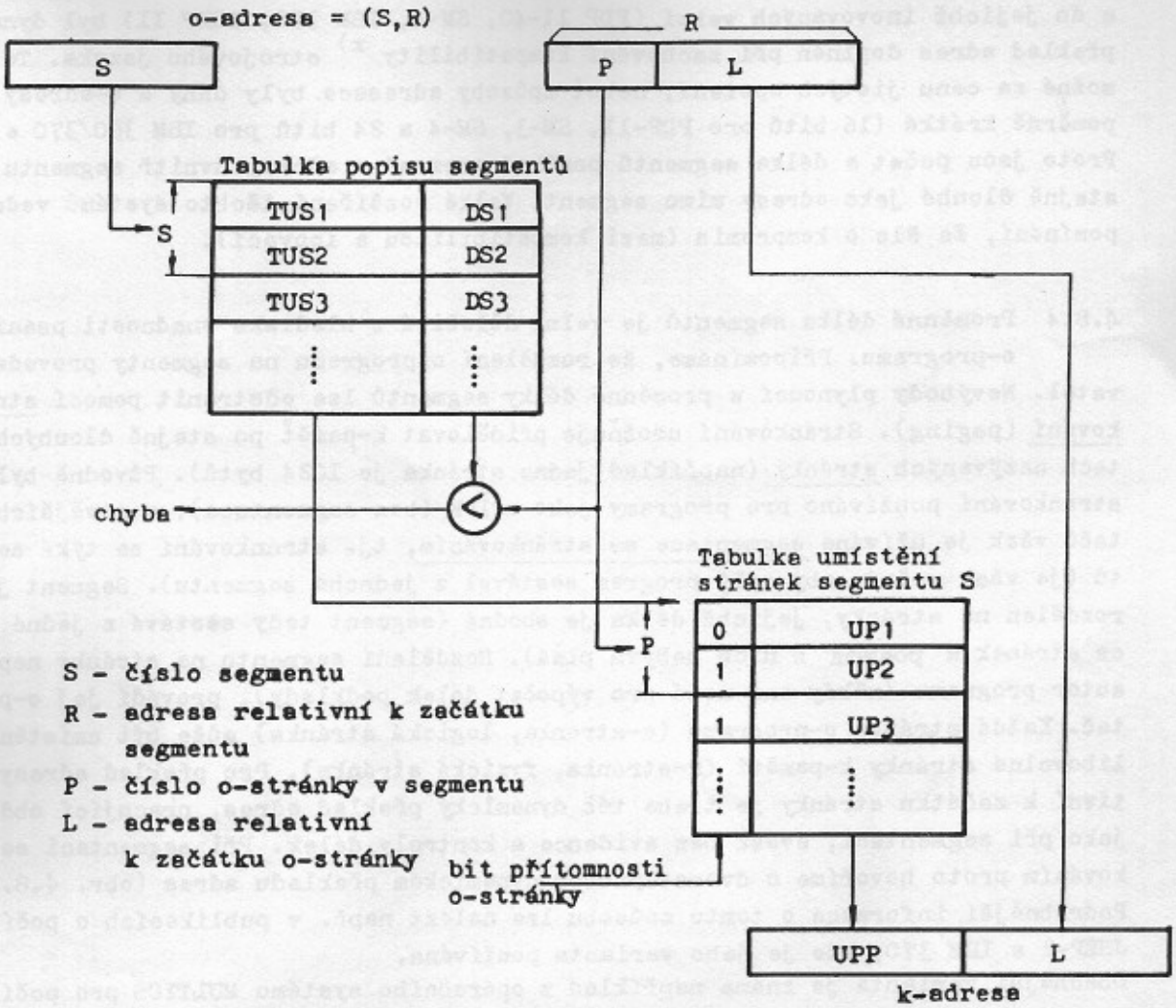
a do jejichž inovovaných verzí (PDP 11-40, SM-4, IBM 370, JSEP II) byl dynamický překlad adres doplněn při zachování kompatibility ±) strojového jazyka. To bylo možné za cenu jistých opezení, neboť způsoby adresace byly dány a o-adresy byly poměrně krátké (16 bitů pro PDP-11, SM-3, SM-4 a 24 bitů pro IBM 360/370 a JSEP). Proto jsou počet a délka segmentů poněkud omezené a adresy uvnitř segmentu jsou stejně dlouhé jako adresy mimo segment. Velké rozšíření těchto systémů vede k zapomínání, že šlo o kompromis (mezi kompatibilitou a inovací).

4.8.4 Proměnná délka segmentů je velmi důležitá z hlediska snadnosti psaní o-programu. Připomínáme, že rozdělení o-programu na segmenty provede uživatel. Nevýhody plynoucí z proměnné délky segmentů lze odstranit pomocí stránkování (paging). Stránkování umožňuje přidělovat k-paměť po stejně dlouhých částech nazývaných stránky (například jedna stránka je 1024 bytů). Původně bylo stránkování používáno pro programy jako celek (bez segmentace), u novějších počítačů však je užívána segmentace se stránkováním, tj. stránkování se týká segmentů (je však možné, aby celý program sestával z jednoho segmentu). Segment je rozdělen na stránky, jejichž délka je shodná (segment tedy sestává z jedné či více stránek a poslední z nich nebývá plná). Rozdělení segmentu na stránky neprovádí autor programu (někdy ani nemá pro výpočet délek podklady), provádí jej o-počítač. Každá stránka o-programu (o-stránka, logická stránka) může být umístěna do libovolné stránky k-paměti (k-stránka, fyzická stránka). Pro překlad adresy relativní k začátku stránky je třeba též dynamický překlad adres, pracující obdobně jako při segmentaci, avšak bez evidence a kontroly délek. Při segmentaci se stránkováním proto hovoříme o dvoustupňovém dynamickém překladu adres (obr. 4.8.2). Podrobnější informace o tomto způsobu lze nalézt např. v publikacích o počítačích JSEP-2 a IBM 370, kde je jeho varianta používána.
Obecnější varianta je známa například z operačního systému MULTICS pro počítač Honeywell 6180.

P o z n á m k a : U počítačů PDP 11/34 a vyšších modelů PDP 11 je používán jednostupňový dynamický překlad adres, který je podobný segmentaci (bez stránkování), popsané v odstavci 4.8.3 (výrobce počítače tento způsob nazýval nejprve segmentace, později jej z nějakých důvodů "přejmenoval" na stránkování). Principiálně stejné řešení je použito u počítačů SM-4.

---

±) slučitelnosti




Obr. 5.8.2 Dvoustupňový dynamický překlad adres

## 4.9 PROCESOR O-POČÍTAČE

4.9.1 Procesor o-počítače (o-procesor) umí provádět příkazy řídícího jazyka a příkazy výkonného jazyka. Příkazy výkonného jazyka jsou uloženy v o-paměti. Operandy užívané těmito příkazy jsou uloženy buď v o-paměti, nebo v registrech o-procesoru (o-registry). Pro vstup a výstup jsou k dispozici o-zařízení, která o-procesor rovněž řídí pomocí příkazů výkonného jazyka.

Jako o-registry jsou obvykle užívány některé registry k-procesoru (k-registry). Stav o- procesoru je určen obsahem o-registrů. Při odnímání a vracení

k-procesoru o-programu je tedy třeba obsah o-registrů zachovat.

Odlišnost o-registrů a k-registrů se projevuje například tím, že

- některé k-registry jsou používány jako o-registry (jsou přístupné v o-programu stejně jako v k-programu anebo jen určitým způsobem);
- některé k-registry jsou přístupné pouze pomocí privilegovaných instrukcí (nejsou přístupné o-programu);
- v jednotlivých režimech činnosti k-procesoru mohou být pod stejným číslem užívány různé k-registry (stejných vlastností).

4.9.2 Zpravidla je implementován jeden o-procesor pro jeden o-počítač. Záleží tedy na stupni multiprogramování a ten se u operačních systémů pro zpracování dávek řídí kritériem optimálního využití k-počítače a u operačních systémů účastnického typu je dán počtem pracujících účastníků.

o-procesor je implementován přidělováním a odnímáním k-procesoru jednotlivým o-procesorům. k-procesor přitom pracuje v režimu "uživatel", jestliže provádí instrukce o-programu, a v režimu "systém", jestliže interpretuje příkazy o-programu (tj. provádí obslužné programy).

Odnímání k-procesoru lze provést při kterémkoli přerušení. Přidělení k-procesoru má smysl provádět o-programům, které nepožádaly o čekání na nějakou událost (třeba na dokončení operace vnějšího zařízení). Z hlediska implementace o-procesoru můžeme tedy rozlišit následující situace:

- o-procesoru je přidělen k-procesor (proces běží),
- o-procesoru není přidělen k-procesor (proces neběží).

Procesem zde rozumíme provádění o-programu, tedy proces úrovně 3. Jestliže proces neběží, rozlišíme následující případy:

- o-procesoru lze přidělit k-procesor (proces připraven),
- o-procesor čeká na událost (proces čeká).

Je tedy potřebné udržovat seznam připravených procesů a dále seznam procesů čekajících na jednu či více událostí. Jestliže událost nastane, je třeba seznamy aktualizovat.

4.9.3 k-procesor lze přidělit jednomu z připravených procesů. Výběr lze provést například podle priority, nebo lze procesy cyklicky střídat (a vynechávat čekající). V obou případech se však může stát, že proces málo používající vnější zařízení běží velmi dlouho, aniž by způsobil přerušení, takže není příležitost k odejmutí procesoru. To má za následek, že činnost vnějších zařízení se zastaví, přestože ostatní procesy mají připraveny pro tato zařízení příkazy. Později může naopak dojít k tomu, že všechny procesy čekají na ukončení operací vnějších zařízení a procesor je tedy nevyužit. Z hlediska efektivnosti využití počítače je potřebné nenechávat ani procesor, ani vnější zařízení zbytečně zahálet.

Aby bylo možné omezit dobu přidělení procesoru jednomu procesu, je užíváno vnější zařízení nazývané časovač (timer). Úkolem časovače je pevně nebo v programem určených intervalech, např. 10 nebo 100 ms, požadovat přerušení. To umož-

ňuje odměřování času a odnímání procesoru jednotlivým procesům (time slicing, doslova odřezávání času). Říká se též, že procesům jsou přidělována časová kvanta (tj. čas přidělení k-procesoru o-procesoru je dávkován). Proces kvantum buď využije, nebo požádá před ukončením kvanta o čekání. Přidělováním časových kvant lze např. řídit dobu odezvy pro jednotlivé účastníky u účastnického operačního systému. Záleží však nejen na velikosti kvanta, ale i na frekvenci přidělování (ne všechna kvanta proces využije plně).

Často se také kombinuje prioritní způsob přidělování s cyklickým, tj. je více cyklických seznamů procesů a je stanovena priorita těchto seznamů. Procesy nejvyšší priority se cyklicky střídají, a jestliže žádný z nich není připraven, dojde na cyklické střídání procesů nižší priority atd.

## 4.10 SEZNÁMENÍ S OPERAČNÍM SYSTÉMEM

Na závěr je třeba upozornit, že v této kapitole je obsažen jen orientační úvod do problematiky operační systémů.

Pro další studium doporučujeme seznámit se pokud možno detailně s používáním některého jednoduššího operačního systému a ujasnit si, jak jsou v tomto konkrétním případě definovány a pojmenovány pojmy používané v této kapitole a jak jsou řešeny problémy, které jsme zde naznačili. Vhodný systém je např. MIKROS.

## Charakteristika operačního systému MS DOS

Operační systém DOS se skládá z řady programů, které provádějí jednotlivé funkce systému. Tyto programy jsou umístěny na diskovém médiu (disketa nebo pevný disk), odkud se podle potřeby zavádějí do paměti počítače.

### 1. Složky operačního systému

Operační systém DOS můžeme rozčlenit na tři části:
- základ systému,
- služební programy,
- překladače.

#### 1.1 Základ systému

Základ operačního systému DOS obsahuje:
- zaváděcí program,
- vazební program (BIO.COM), jehož prostřednictvím se uskutečňuje styk s technickým vybavením,
- jádro systému DOS uložené v souboru DOS.COM,
- procesor příkazů obsluhy (COMMAND.COM).

Zaváděcí program slouží k počátečnímu zavedení operačního systému do paměti a k zahájení jeho činnosti. Tento program je umístěn na začátku každé diskety (povrch 0, sektor 1) s formátem operačního systému DOS. Na pevném disku je zaváděcí program umístěn v prvním sektoru oblasti určené pro systém DOS.

Vazební program na technické vybavení je uložen v souboru BIO.COM na systémovém médiu. Při zavádění systému se tento program uloží do paměti, kde zůstane až do ukončení práce systému. Při práci systému zprostředkovává spojení mezi operačním systémem a technickým vybavením počítače. Ze strany technického vybavení je přitom komunikace zajišťována programem BIOS (Basic Input/Output Systém), který je uložen v paměti ROM. Do této paměti nelze zapisovat, lze ji pouze číst. V programu BIOS jsou realizovány základní vstupní a výstupní operace. Tento program je plně závislý na technickém vybavení.

Jádro systému (DOS) je uloženo na systémovém médiu v souboru DOS.COM, odkud je při zavedení systému přeneseno do paměti. Tam zůstane uloženo až do ukončení práce systému. Zatímco programy BIO.COM a BIOS zajišťují fyzický přístup k technickému vybavení, DOS určuje logický průběh zpracování. Řídí provádění vstupních a výstupních operací na logické

úrovni, práci s adresářem, přidělování místa na disku, správu paměti pro zápis a čtení (RWM-RAM) atd. Soubor DOS.COM, je tzv. skrytý soubor. To znamená, že se ve výpisu adresáře disku příkazem DIR vynechává.

Procesor příkazů obsluh uložený v souboru COMMAND.COM, zajišťuje spojení mezi uživatelem a systémem. Jeho hlavní funkcí je provádění příkazů obsluhy. Sestává z rezidentní, inicializační a tranzientní části.

Rezidentní část je po svém zavedení trvale přítomná v paměti a zajišťuje zpracování některých přerušení (např. klíče pro zastavení programu), zavádění tranzientní části procesoru do paměti, zpracování chyb atd.

Inicializační část se používá při počátečním zavedení systému a slouží např. k určení adresy, od níž mohou být ukládány uživatelské programy. Při zavedení prvního programu se inicializační část přepíše, protože již není potřebná.

Tranzientní část se zavádí do horní oblasti paměti, kde může být v případě potřeby přepsána služebním nebo uživatelským programem. Slouží k vlastnímu provádění příkazů obsluhy zadaných na terminálu nebo získaných z příkazového souboru. Tyto příkazy provádí buď přímo nebo zavede a spustí jiný potřebný program. Podle toho, zda jsou jednotlivé příkazy obsluhy realizovány přímo v procesoru příkazů nebo samostatnými programy, rozdělujeme je na interní a externí.

## 1.2 Služební programy

Služební programy slouží k provádění různých funkcí, které uživateli usnadňují práci s počítačem.

Systémový disk obsahuje skupinu služebních programů, které slouží k provádění externích příkazů obsluhy. Zpracování těchto příkazů je řízeno procesorem příkazů obsluhy, který vypíše na terminál nápovědnou zprávu, přijme příkaz z klávesnice a vyvolá příslušný služební program. Po provedení požadované funkce opět vypíše nápovědnou zprávu. Mezi externí příkazy obsluhy patří např. příkaz FORMAT pro formátování disket.

Další skupinou služebních programů jsou programy, jejichž činnost se řídí pomocí jejich vlastních příkazů. Sem patří např. řádkový editor EDLIN.

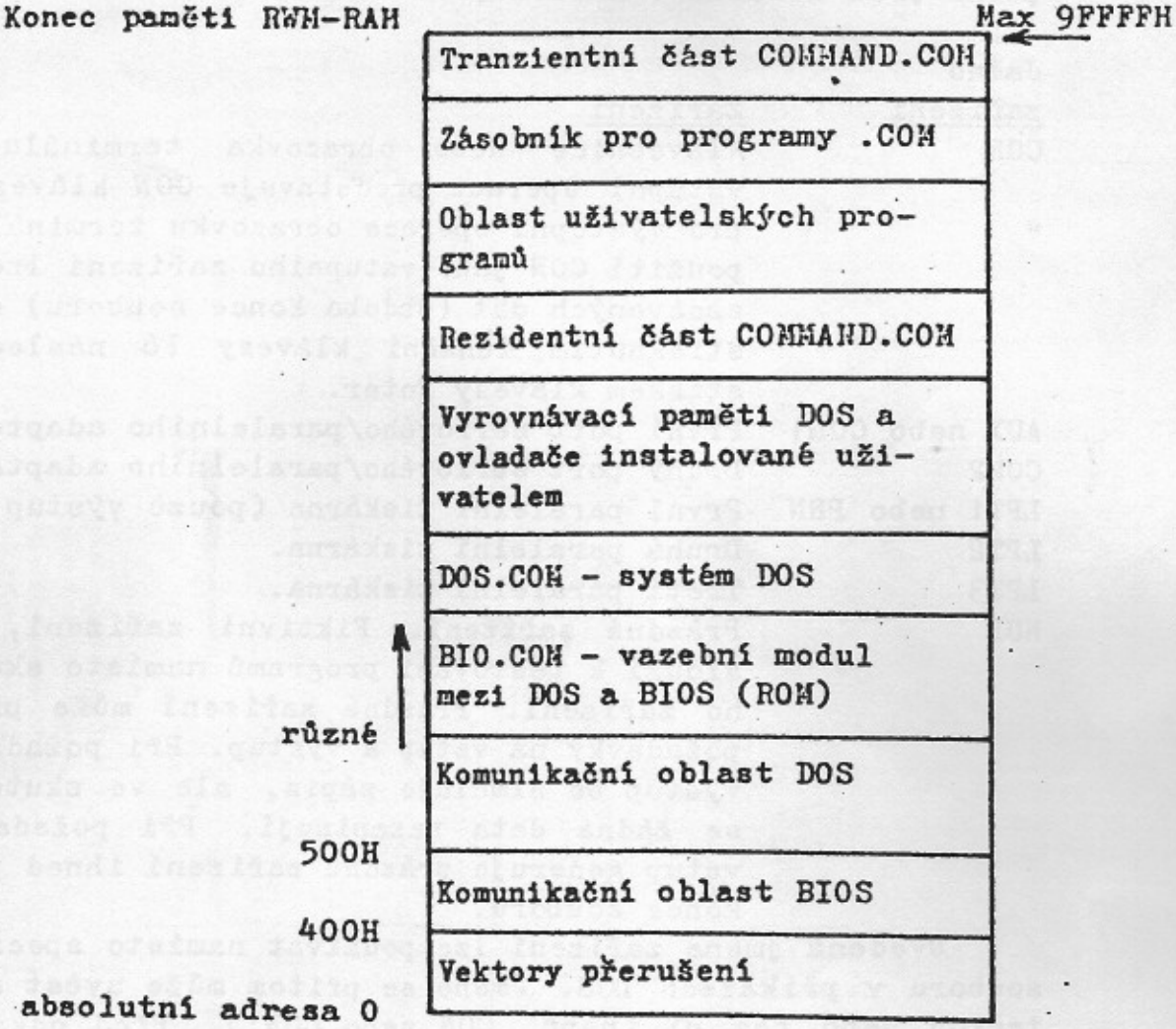
## 1.3 Překladače

Základními programovacími jazyky v operačním systému DOS jsou Basic a makroasembler. K dispozici jsou i další

programovací jazyky.

## 2. Obsazení paměti

Na následujícím obrázku je znázorněno obsazení paměti RWM-RAM při práci operačního systému DOS.




Moduly operačního systému DOS jsou umístěny vždy ve stejné oblasti ve spodní části paměti, takže při změně velikosti paměti se neposouvají. Uživatelským programům se přiděluje paměť za rezidentní částí COMMAND.COM. Uživatelský program může v případě potřeby používat i tu část paměti, do níž se ukládá tranzientní část COMMAND.COM.

Pro oblast obrazové paměti RWM-RAM (VIDEO RAM) je vyhražen adresový prostor 0A0000H až 0BFFFFH. Ve zbývající části adresního prostoru mikroprocesoru 8086, tj. od adresy 0C0000H do adresy 0FFFFFH jsou paměti typu ROM.

## Nesouborově orientovaná zařízení

Dalšími zařízeními, pomocí kterých lze provádět vstup a výstup dat, jsou nesouborově orientovaná zařízení. Na těchto zařízeních se nepracuje se soubory, ale vždy pouze s daným zařízením. Patří sem zejména terminál (klávesnice s obrazovkou) a tiskárna. Na nesouborová zařízení se uživatel odkazuje pomocí speciálních jmen definovaných systémem DOS. Tato jména jsou uvedena v následující tabulce.

| Jméno zařízení | Zařízení |
|---|---|
| CON | Klávesnice nebo obrazovka terminálu. Pro vstupní operace představuje CON klávesnici a pro výstupní operace obrazovku terminálu. Při použití CON jako vstupního zařízení lze konec zadávaných dat (obdoba konce souboru) oznámit stisknutím funkční klávesy F6 následovaným stiskem klávesy Enter. |
| AUX nebo COM1 | První port sériového/paralelního adapteru. |
| COM2 | Druhý port sériového/paralelního adapteru. |
| LPT1 nebo PRN | První paralelní tiskárna (pouze výstup dat). |
| LPT2 | Druhá paralelní tiskárna. |
| LPT3 | Třetí paralelní tiskárna. |
| NUL | Prázdné zařízení. Fiktivní zařízení, které slouží k testování programů namísto skutečného zařízení. Prázdné zařízení může přijímat požadavky na vstup a výstup. Při požadavku na výstup se simuluje zápis, ale ve skutečnosti se žádná data nezapisují. Při požadavku na vstup generuje prázdné zařízení ihned příznak konce souboru. |

Uvedená jména zařízení lze používat namísto specifikace souboru v příkazech DOS. Jméno se přitom může uvést s dvojtečkou nebo bez ní (např. CON nebo CON:). Před odkazem na určité zařízení je potřeba se ujistit, že je skutečně k počítači připojeno. Pokus o provedení operace na neexistujícím zařízení může způsobit nepředvídané chyby.

## 3. SOUBORY

Data a programy se uchovávají na discích v souborech. Každý disk obsahuje adresář souborů, které jsou na něm uloženy a tabulku sektorů obsazených soubory. Tato tabulka je z důvodu větší bezpečnosti uložení dat obsažena na disku dvakrát. Místo na disku se souborům přiděluje po alokačních blocích sestávajících z jednoho nebo více sektorů.