7
PAMÄŤOVÝ PODSYSTÉM ČÍSLICOVÉHO POČÍTAČA

7.1 Hierarchická štruktúra pamäte

Jedným zo základných komponentov štruktúrnej organizácie číslicového počítača (ČP) je jeho *pamäť*. Úlohou pamäte je umožniť zápis, pamätanie (uloženie) a čítanie kódov inštrukcií, údajov, medzivýsledkov a výsledkov operácií, stavy ČP a jeho podsystémov a kódy iných podporných programov, ktorými sú definované procesy spracovania a komunikácií informácií v ČP. Vzhľadom na nevyhnutnosť vyriešenia konfliktu medzi požiadavkami na hodnoty základných parametrov pamäte, ktorými sú kapacita, prístupová doba a cena, používa sa niekoľko druhov pamätí s rozličnými hodnotami uvedených parametrov. Zo systémového hľadiska predstavujú tieto pamäte jediný celok, vytvárajúc *pamäťový podsystém* ČP [67], [84], [98], [121], [161].

7.1.1 Klasifikácia pamätí

Podľa spôsobu sprístupňovania informácií pri ich čítaní/zápise z/do pamäte (pri aktivácii pamäte), ktorý vyplýva z konštrukcie a štruktúrnej organizácie pamäte sa rozlišujú [121]:

1. Pamäte s postupným prístupom (*SAM* - *S*equential *A*ccess *M*emory).
2. Pamäte s priamym prístupom (*DAM* - *D*irect *A*ccess *M*emory) s komerčným označením:
   - *RAM* (*R*andom *A*ccess *M*emory), v priamom prístupe ku ktorým sa zvýrazňuje prvok náhodilosti,
   - *RWM* (*R*ead *W*rite *M*emory), pri ktorých sa zvýrazňuje ich určenie na čítanie a zápis (charakteristické i pre pamäte RAM),
   - *ROM* (*R*ead *O*nly *M*emory) - permanentné pamäte, ktoré umožňujú iba čítanie ich informačného obsahu, ktorý dodáva výrobca,
   - *PROM* (*P*rogramable *ROM*) - permanentné pamäte, ktoré umožňujú iba čítanie ich informačného obsahu, ktorý jednorázovo môže programovať používateľ,
   - *REPROM* (*R*eprogramable *PROM*) - semipermanentné pamäte, ktoré umožňujú iba čítanie ich informačného obsahu, ktorý môže používateľ viacnásobne preprogramovať.
3. Pamäte s asociatívnym prístupom (*CAM* - *C*ontent *A*ccess *M*emory), ktoré sprístupňujú ich obsah podľa výberového kľúča.

Na základe pamätanej informácie vo vzťahu k pamäťovému nosiču sa rozlišujú:

- *statické pamäte*, na báze prvkov z dvoma stabilnými stavmi (*SRAM – S*tatic *RAM*),
- *dynamické pamäte*, na báze časovo ohraničenej reprezentácie bitových hodnôt, ktoré je nutné v pravidelných intervaloch obnovovať (*DRAM - D*ynamic *RAM*).

7.1.2 Požiadavky na organizáciu pamäte

Hlavnou požiadavkou na pamäte ČP je ich dostatočne veľká kapacita a malá prístupová doba. Tieto požiadavky sú protichodné a riešia sa hierarchickým usporiadaním pamäťového podsystému (Kap.7.1.3). Okrem uvedeného, architektúra ČP definuje pre činnosť pamäťového podsystému rad zložitejších požiadaviek, ktoré vyplývajú z vlastností použitých programových prostriedkov a ktoré sú určované osobitnosťami ČP, orientovanými na použitie týchto prostriedkov. Z uvedeného pohľadu medzi najdôležitejšie požiadavky na *organizáciu pamäťového podsystému* patria tie, ktoré vyplývajú zo [67]:

- štruktúry údajov,
- mechanizmu ochrany pamäte,
- systému mikroprogramového riadenia,
- technologického riešenia a spôsobu komunikácie s procesorom.

Hlavná pamäť je obyčajne reprezentovaná lineárnou postupnosťou pamäťových miest (buniek), ktorá sa sprístupňujú jednorozmerným adresovateľným priestorom. Takýto spôsob priameho adresovania ohraničuje dosahovanie optimálnych parametrov, reprezentujúcich celkovú výkonnosť pamäťového podsystému (Kap.7.6). Preto sa uplatňujú rôzne prístupy nepriameho adresovania (Kap.2.2), ktoré umožňujú efektívne adresovať informácie uložené v pamäťovom podsystéme (napríklad znížiť prístupovú dobu k pamäti a pod.). Lineárna organizácia adresovania negatívne ovplyvňuje efektívnu reprezentáciu mnohých často používaných údajových štruktúr (viacrozmerné polia, matice, zásobníky, tabuľky, stromy a pod.). Z týchto dôvodov sa organizácia pamäte prispôsobuje uvedeným údajovým štruktúram. Príkladom realizácie takéhoto prístupu sú organizácie zásobníkových, vyrovnávacích a asociatívnych pamätí, mnohodimenzionálnych pamätí so šikmým prístupom a pod.

V *multiprogramových počítačových systémoch* je potrebné presne definovať možnosť prístupu jednotlivých programov k pamäti a tým aj vylúčiť ich nepovolenú a nežiadúcu aktiváciu. Aby sa vylúčili nepovolené prístupy k chráneným oblastiam pamäte, používajú sa špeciálne programové a technické prostriedky, ktoré sú charakteristické pre organizáciu pamäte a jej riadenie, vo všeobecnosti označované ako *ochrana pamäte*.

V triede *špecializovaných ČP* vystupujú do popredia výpočtové systémy, ktoré sú orientované na problémovo orientované spracovanie informácií (sieťové počítačové systémy, databázové systémy, signálové procesory, znalostné počítače, neurónové siete a pod.). V týchto systémoch spravidla musí pamäťový podsystém čo najtesnejšie spolupracovať s procesorom, ktorá požiadavka vyúsťuje až do zlúčenia funkcie pamätania so spracovaním informácií. Jedným z druhov takýchto pamätí je *asociatívna pamäť*, v ktorej sa pamäťové miesta sprístupňujú na základe porovnávania príznakov umiestnených v časti ich informačných príznakov, *procesory na pamäťových čipoch* a pod.

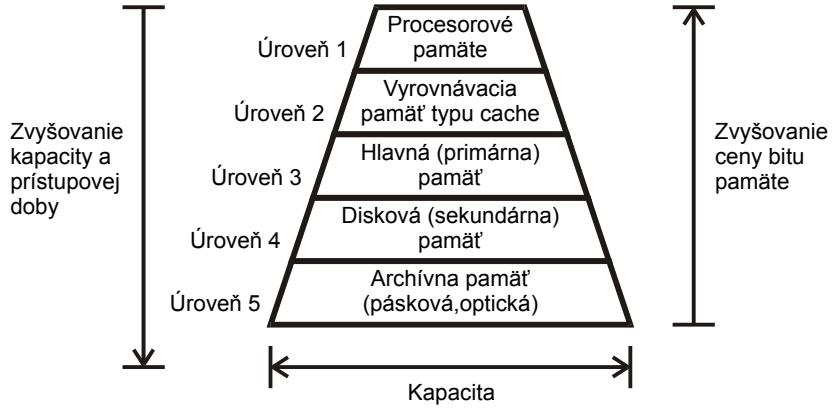
V počítačových architektúrach s mikroprogramovým riadením, pri ktorom sa strojové inštrukcie realizujú na základe vykonania postupnosti elementárnych inštrukcií – mikrooperácií, definovaných príslušným súborom mikroinštrukcií, vznikajú požiadavky na organizáciu tzv. riadiacich pamätí. V nich sú uložené postupnosti kódov mikroinštrukcií – mikroprogramov, zodpovedajúcich príslušným kódom inštrukcií. V špecializovaných ČP a počítačoch HLL (Kap.2.2), riadiace pamäte tvoria významnú časť technických prostriedkov ich procesorov. Koncepcia riadiacich pamätí je využívaná v celom rade aplikácií, v ktorých sa často používa na tvorbu používateľsky orientovaných mikroprogramov, nových typov inštrukcií a reprezentácie objektov definovaných používateľom.

S nepretržitým zvyšovaním kapacity pamätí vystupujú v súčasnosti do popredia koncepcie pamätí s rozšírením ich funkcie prostredníctvom implementácie ďalších obvodov na pamäťovom

čipe s cieľom zvýšenia výkonnosti organizácie riadenia pamäťového podsystému. Patria medzi ne *pamäte s procesorom* (*PIM – P*rocessor-*i*n-*m*emory) alebo *inteligentné RAM* (*IRAM*), ktoré na svojom čipe integrujú jeden alebo viac procesorov, ktoré sa vyznačujú efektívnym zosúladením prenosových rýchlostí procesora a pamäťového podsystému.

7.1.3 Mnohoúrovňový pamäťový podsystém

Pamäťový podsystém ČP je charakterizovaný celým radom protichodných požiadaviek, z ktorých najmä požiadavky na veľkú kapacitu, malú prístupovú, resp. vybavovaciu dobu a nízku cenu technických prostriedkov nemožno realizovať jedným druhom pamäte. To je dôvod na vznik kompromisného riešenia organizácie pamäťového podsystému, ktoré spočíva v hierarchickom usporiadaní jeho štruktúry [98], [120].



Obr.7.1. Koncepcia mnohoúrovňového pamäťového podsystému číslicového počítača.

Prvú úroveň pamäťového podsystému reprezentujú veľmi rýchle *procesorové pamäte* s malou kapacitou, vykonávajúcich rôzne druhy pamäťových funkcií v štruktúre procesora, v dôsledku čoho reprezentujú základný druh tzv. vnútorných pamätí ČP lokalizovaných na čipe procesora.

Druhú úroveň pamäťového podsystém tvoria vyrovnávacie pamäte, ktorých funkciou vo všeobecnosti je vyrovnávanie rýchlosti prenosu údajov medzi dvoma prvkami s rozdielnou rýchlosťou spracovania informácií. Pamäť určená na vyrovnávanie rýchlostí prenosu údajov medzi procesorom a hlavnou pamäťou sa nazýva *vyrovnávacia pamäť typu „cache“* (*pamäť cache*). V štruktúre pamäťového podsystému súčasných ČP sú pamäte cache usporiadané v jednej až troch úrovniach (L1, L2, L3), pričom môžu tvoriť samostatné pamäťové čipy, alebo sú časťou procesorového čipu (spravidla pamäť cache úrovne L1). Jednotlivé úrovne pamätí cache sa odlišujú hodnotami svojich základných parametrov (príklad odlišnosti parametrov úrovne L1 a L2 je uvedený v Tab.7.1) a organizáciou ich sprístupňovania (Kap.7.3).

Treťou úrovňou pamäťového podsystému je *hlavná pamäť*, ktorá tvorí jeho jadro. Je v nej uložený program a údaje, ktoré v danom časovom intervale (behu programu) reprezentujú súčasť procesu spracovania informácií. Hlavná pamäť a vyrovnávacie pamäte predstavujú spravidla *fyzický adresovateľný priestor* v *systéme virtuálnej pamäte* ČP (Kap.7.5).

Štvrtou úrovňou pamäťového podsystému sú *veľkokapacitné vonkajšie pamäte* organizované ako *hromadné*, resp. *sekundárne pamäte* na rôznych typoch magnetických a optických diskoch, často označovaných ako *diskové pamäte typu CD* (*C*ompact *D*isc) alebo *DVD* (*D*igital *V*ersatil *D*isc).

Piatu úroveň pamäťového podsystému predstavujú *archívne pamäte* charakteristické pre výpočtové prostredia špecializovaných aplikácií využívajúce vysokovýkonné a veľkokapacitné počítačové a komunikačné systémy a superpočítače.

Charakteristické parametre pamätí v jednotlivých úrovniach pamäťového podsystému ČP sú uvedené v Tab.7.1. Pre mnohoúrovňový pamäťový podsystém (Obr.7.1) vo všeobecnosti platí, že v smere od procesora ku archívnej pamäti sa kapacita jednotlivých čiastkových pamäťových úrovní, vyjadrená v MB, resp. TB, zväčšuje a prístupová, resp. vybavovacia doba, vyjadrená v milisekundách až minútach (archívne pamäte) a cena jedného bitu informácií sa v opačnom poradí znižuje.

Tab.7.1.Charakteristiky pamätí v jednotlivých úrovniach pamäťového systému

Úroveň Cha- rakteristika	Úroveň 1 CPU (Registre)	Úroveň 2 Cache (L1/L2)	Úroveň 3 Hlavná pamäť	Úroveň 4 Disková pamäť	Úroveň 5 Archívna pamäť
Technológia	ECL	čip 512 kB SRAM	čip 128 MB DRAM	Diskové jednotky	Magnetické páskové jednotky
Prístupový čas, t_i	< 2 ns	< 5 ns	< 100 ns	< 10 ms	2 - 20 min
Kapacita pamäte, c_i	512 B	128 kB / 4MB	32 - 512 MB	10 - 100 GB	10 TB
Prenosová rýchlosť, p_i	4k - 32k MB/s	800 - 5000 MB/s	400 - 2000 MB/s	4 -32 MB/s	100 -1000 kB/s
Prenosová jednotka, x_i	4 - 8 B/slovo	32 B/blok	1 KB/stránka	5 – 512 KB/súbor	Záložná pamäť
Riadenie pamäte	Kompilátor	Hardvér	Operačný systém (OS)	OS/ používateľ	OS/ používateľ
Zálohovanie	Cache	Hlavná pamäť	Disková pamäť	Pásková pamäť	–

7.2 Procesorové pamäte

Procesorové pamäte (*PP*) predstavujú rôzne typy registrových pamätí, ktoré z funkčného hľadiska majú rôznu funkčnú a štruktúrnu organizáciu a technologickú realizáciu [67], [121]. Tieto pamäte sú zväčša súčasťou procesora, môžu byť však aplikované i v iných jednotkách ČP. Do skupiny procesorových pamätí patria i rôzne typy permanentných, semipermanentných pamätí, ktoré zahrňujú i širokú triedu programovateľných zákaznických a polozákaznických obvodov.

7.2.1 Permanentné a semipermanentné pamäte

Permanentné pamäte predstavujú pamäte s nemeniteľným obsahom (pamäte ROM – *R*ead *O*nly *M*emory), z ktorých sa pri prevádzke údaje iba čítajú. Ich obsah je dodaný výrobcom pamäti ROM alebo je jednorázovo programovaný používateľom na výrobcom dodaný polotovar (*PROM* – *P*rogrammable *ROM*) prostredníctvom osobitného programovacieho zariadenia. Pamäte ROM, ktoré umožňujú ich viacnásobné preprogramovanie používateľom sa nazývajú *semipermanentné* (*EPROM – E*xternal reprogrammable *PROM*).

Uvedené typy pamätí sa využívajú vo väčšine prípadov ako pamäťové obvody v rôznych typoch riadiacich častí číslicových systémov, ktoré v tomto prípade sú procesory ČP. Pamäte s nemeniteľným obsahom sú najčastejšie reprezentované prostredníctvom pamätí typu ROM, PLA, GA a pod. vo funkcii:

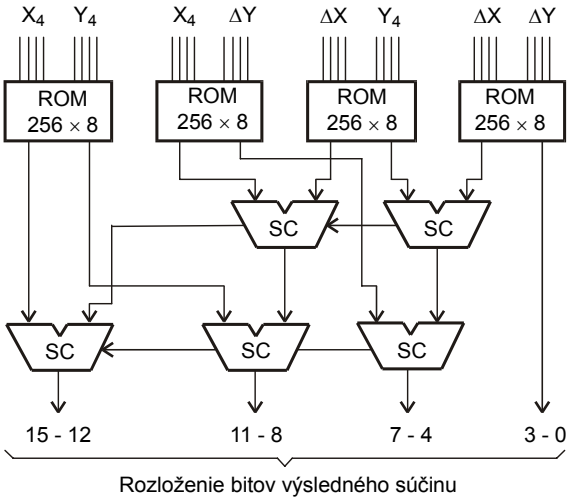
- generátora logických funkcií (pamäť pravdivostnej tabuľky),
- generátora kódu (prevodník kódu),
- generátora znakov (pamäť bodov matice rastra znakov),

- generátora funkcií (pamäť tabuliek hodnôt funkcie),
- špecializovanej ALJ (pamäť výsledkov operácií),
- riadiacej pamäte (pamäť riadiacich signálov v riadiacich jednotkách a mikroprogramových automatoch).

Tak napríklad, pri realizácii logických funkcií pamäťou ROM sa prostredníctvom adresových vodičov zobrazujú hodnoty premenných logickej funkcie a prostredníctvom bitových vodičov, hodnoty príslušných logických funkcií. Znamená to, že v pamäťovej matici sa v podstate realizuje *pravdivostná tabuľka* vyžadovaných funkcií, zatiaľ čo konkrétne hodnoty premenných tejto funkcie definujú *adresu príslušného riadku* pamäte ROM. Prednosťou realizácie logických funkcií pamäťou ROM je 10-krát nižšia cena v porovnaní s cenou ich obvodovej realizácie, jednoduchší návrh a rovnaké časové oneskorenie pre každú logickú funkciu. Za nevýhodu treba považovať značnú pamäťovú kapacitu pre väčší počet vstupných premenných. Úspornejšie riešenie možno dosiahnuť prostredníctvom *programovateľných logických polí PLA* (*P*rogrammable *L*ogic *A*rray), alebo *programovateľných hradlových polí FPGA* (*F*ield *P*rogrammable *G*ate *A*rray).

Aplikácia pamäte ROM vo funkcii *generátora kódu* spočíva v tom, že v závislosti od vstupnej informácie, ktorá sa privedie k jej adresovému vstupu, generuje na bitových vodičoch vyžadované binárne reťazce. Podobne plní aj úlohu *prevodníka kódov*.

Jednou z najrozšírenejších aplikácií pamäte ROM je jej využitie v *mikroprogramových systémoch*, kde vykonáva funkciu *riadiacej pamäte*. Uložený obsah riadiacej pamäte sa interpretuje ako súbor kódov *mikroinštrukcií*, ktoré inicializujú realizáciu *elementárnych operácií* (*mikrooperácií*) v systéme s mikroprogramovým riadením.



Obr. 7.2. Aplikácia pamäte ROM pri realizácii operácie násobenia

Veľmi dôležitou aplikáciou pamäte ROM je vytváranie tabuliek funkcií (napríklad, trigonometrických, logaritmických a pod.). Spôsob ich programovania závisí od vyžadovanej presnosti Napríklad, do pamäte ROM s kapacitou 128×8 bitov možno zaznamenať 128 hodnôt funkcie sin x pre x = 0^0 až 90^0 v prírastkoch po 0.7^0, pričom osembitový výstup umožňuje vyjadriť príslušné hodnoty sin x s presnosťou 99.9^0. V súvislosti s funkciou pamäte ROM ako tabuľky hodnôt do popredia vystupuje jej využitie tabuľkového zápisu výsledkov určitej aritmetickej operácie (najčastejšie celočíselnej) so všetkými možnými kombináciami jej operandov. Napríklad, ak na zostavenie tabuľky násobenia dvoch 4-bitových celých čísiel je potrebná pamäť ROM s kapacitou $2^8 \times 8 = 2048$ bitov, tak pri 8-bitových celých číslach sa vyžaduje už kapacita $2^{16} \times 16$ bitov. Vhodným usporiadaním štruktúry pamätí ROM s malou kapacitou je možné rozsiahlu

tabuľku celočíselných výsledkov realizovať s podstatne menšou bitovou kapacitou. Naznačený prístup aplikácie pamätí ROM v úlohe špecializovaných aritmetických jednotiek je v ďalšom ilustrovaný na príklade násobenia dvoch celých 8-bitových čísiel X a Y, zobrazených v tvare:

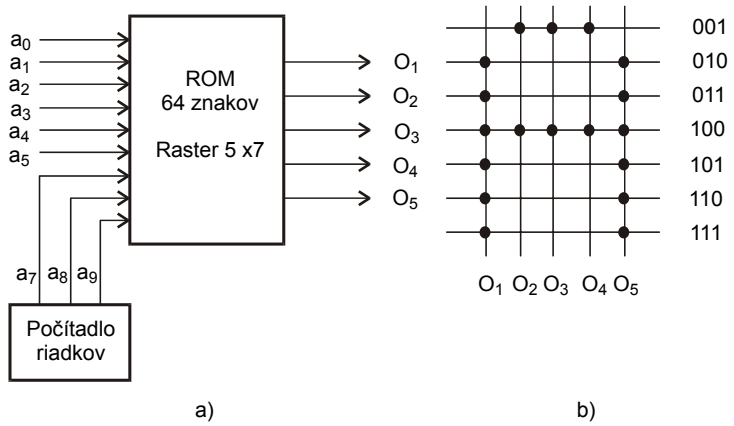
$$X = \tilde{x}_8\tilde{x}_7\tilde{x}_6\tilde{x}_5\,0\,0\,0\,0 + 0\,0\,0\,0\;\tilde{x}_4\tilde{x}_3\tilde{x}_2\tilde{x}_1;\quad \tilde{x}_i \in \{0,1\}$$
$$Y = \tilde{y}_8\tilde{y}_7\tilde{y}_6\tilde{y}_5\,0\,0\,0\,0 + 0\,0\,0\,0\,\tilde{y}_4\tilde{y}_3\tilde{y}_2\tilde{y}_1;\quad \tilde{y}_i \in \{0,1\}$$

Ak prvá časť čísiel X, Y, obsahujúca číslice najvyšších rádov sa označí X_4 a Y_4 a druhá časť týchto čísiel obsahujúca číslice najnižších rádov sa označí ΔX a ΔY, výsledkom ich násobenia upravených do uvedených tvarov je:

$$X \times Y = (X_4 + \Delta X)\,(Y_4 + \Delta Y) = X_4Y_4 + X_4\Delta Y + Y_4\Delta X + \Delta X\,\Delta Y$$

Tieto súčiny sa môžu realizovať pamäťami ROM s kapacitou 2048 bitov (každá s piatimi štvorbitovými sčítačkami) tak, ako je to uvedené na Obr. 7.2.

Ďalšou aplikačnou oblasťou pamäte ROM je generovanie abecedno-číslicových znakov, ktoré sa môžu použiť napríklad pre zobrazovacie jednotky. Základný princíp pre túto aplikáciu pri zobrazovaní znakov v matici 7 × 5 je uvedený na Obr. 7.3. Napríklad, vyžadovaný znak „A“ je určený vstupnou adresou A_0 až A_5, prostredníctvom ktorej je zakódovaných 64 znakov. Bodmi v rastri matice príslušného znaku je zakódovaný jeho obrys, ktorý sa sníma postupným čítaním riadkov adresovaných vodičmi A_6, A_7, a A_8. V danom prípade ide o horizontále snímanie znakov, ktoré predstavuje základný princíp obvodového riešenia generátora znakov v zobrazovacích jednotkách.



Obr.7.3. Princíp aplikácie pamäte ROM v generátore znakov (a – adresovanie obsahu pamäte, b – interpretácia uložených slov: znak „A“).

7.2.2 Zápisníková pamäť

Štruktúra *zápisníkovej pamäte* (*ZP*) [67], [161], určená najčastejšie na krátkodobé ukladanie a výber hodnôt operandov a výsledkov operácií a organizovaná ako registrová pamäť je na Obr. 7.4. Každý *register* RG_i (i = 1, 2, ..., m) predstavuje pamäťovú bunku zápisníkovej pamäte, ktorej *adresa ADR* po zápise v *registri adresy RGA* sa dekóduje v *dekodéri adresy DKA*. Výstupy DKA hradlujú vstupy alebo výstupy jednotlivých registrov.

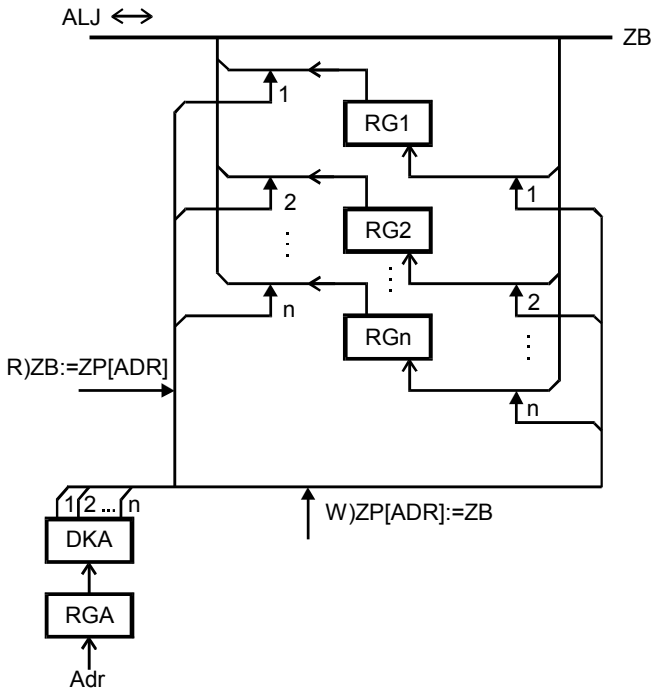
Charakteristickými operáciami ZP je prenos údajov medzi jej registrami, určenými adresou a inštrukciou definovanými registrami ALJ, resp. procesora, čo sa formálne vyjadrí v tvare:

R) *A* := *ZP* [*ADR*]

pri čítaní údajov zo *ZP* a

W) *ZP* [*ADR*] := *A*

pri zápise údajov do *ZP*, kde *R* a *W* sú riadiace signály, inicializujúce registrovú operáciu *čítania* a *zápisu*.



Obr.7.4. Štruktúrna schéma zápisníkovej pamäte registrového typu.

7.2.3 Zásobníková pamäť

Zásobníková pamäť (*ZS*) uskutočňuje zápis a čítanie v inverznom frontovom režime *LIFO* (*L*ast *I*n *F*irst *O*ut). Základným prvkom ZS je počítadlo adries vo funkcii ukazovateľa adresy jeho pamäťovej bunky (registra), tzv. *smerník zásobníka* (*SM*). Priebežný stav SM je dekódovaný dekodérom adries DKA, čím sa realizuje výber príslušnej položky (registra) v ZS. Štruktúrna schéma ZS je uvedená na Obr. 7.5 [121], [161].

Východiskový stav ZS sa nastaví mikrooperáciou R_0) *SM* : = *0*, v dôsledku realizácie ktorej je stavom Zápis údajov do *ZS* z jej údajového vstupu (vstupnej zbernice) *DI* sa uskutoční v priebehu dvoch krokov, z ktorých druhý je určený na inkrementáciu smerníka *SM* zásobníka s oneskorením τ.

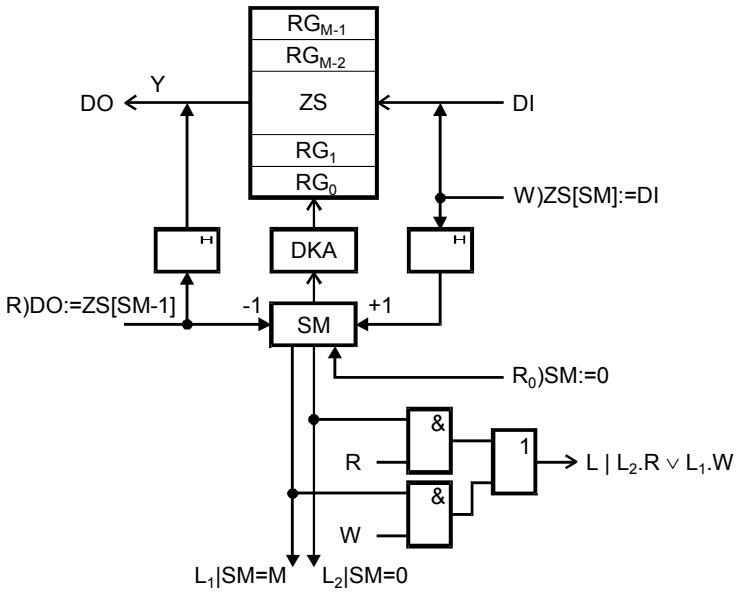
W) *ZS* [*SM*] := *DI*
W(τ)) *SM* := *SM* + 1

Pri zápise postupnosti údajov, pripravených na vstupe ZS, sa tieto umiestňujú v lineárnej postupnosti registrov zásobníka *ZS* [*S*], *ZS* [*S* + 1], *ZS* [*S* + 2] atď., kde S je stav SM v okamihu začiatku zápisu.

Čítanie údajov zo ZS do jeho údajového vstupu (výstupnej zbernice) *DO* sa uskutoční taktiež v priebehu dvoch krokov, z ktorých v prvom sa uskutočňuje dekrementovanie smerníka *SM* zásobníka a až v druhom, s oneskorením τ, sa vykoná vlastné čítanie:

$R)\ SM := SM - 1$
$R(\tau))\ DO := ZS\ [SM]$

Znamená to, že ak napríklad údaje na vstupe ZS boli zapísané v poradí A, B, C, potom môžu byť prečítané iba v opačnom poradí, t.j. v poradí C, B, A.



Obr.7.5. Štruktúrna schéma zásobníkovej pamäte.

Správna funkcia ZS je indikovaná stavovými premennými L_1, L_2, a L_3 nasledovne: L_1– ZS zaplnený, L_2 – ZS prázdny, L_3 – nedovolená aktivácia ZS (požiadavka na zápis do zaplneného ZS alebo na čítanie z prázdneho ZS), t.j.:

$L_1|\ SM = 0$
$L_2|\ SM = M$
$L_3|\ (L_1 \wedge W) \vee (L_2 \wedge R)$

. Typické aplikácie ZS vyplývajú z bezadresového sprístupňovania jeho položiek, ktoré nachádza významné uplatnenie najmä pri:

- *výpočte aritmetických výrazov* zobrazených v postfixnom bezzátvorkovom tvare (pre účely syntaktickej analýzy a konštrukcie prekladačov),
- *organizovanie prerušenia* a *volanie podprogramov* (uloženie návratových adries a stavov prerušeného programu pri prechode do príslušného obslužného programu resp. podprogramu, ukladanie globálnych a lokálnych parametrov procedúr).

Výpočet zložitých aritmetických výrazov podporovaný zásobníkom sa vo všeobecnosti programuje v priebehu dvoch etáp. Počas prvej etapy sa východiskový výraz transformuje do bezzátvorkového tvaru. V tomto tvare postupnosť operandov a operačných znakov zodpovedá vykonaniu operácií v ZS. Počas druhej etapy sa každému operačnému znaku vo výraze v bezzátvorkovom tvare pridelí príslušná bezadresová zásobníková inštrukcia. Napríklad, aritmetickému výrazu

$$Y := A \times ((C + D) / (E \times F + H) - K)$$

zodpovedá nasledujúci bezzátvorkový tvar

$$AB \times CD + EF \times H + / K - \times Y$$

Program, zodpovedajúci tomuto rozkladu je uvedený v Tab. 7.2. Symbol ∗ označuje stav SM v okamihu ukončenia operácie. Z uvedeného príkladu vidno, že počet aktivácií pamäte, z ktorej sa sprístupňujú operandy, sa rovná počtu premenných v aritmetickom výraze. Všetky medzivýsledky sa automaticky uchovávajú v ZS, čím sa znižuje celkový počet aktivácií pamäte, v dôsledku čoho sa výrazne zvyšuje operačná rýchlosť procesora pri vykonávaní aritmetických operácií.

Tab.7.2. Výpočet aritmetického výrazu programovaním zásobníkovej pamäte.

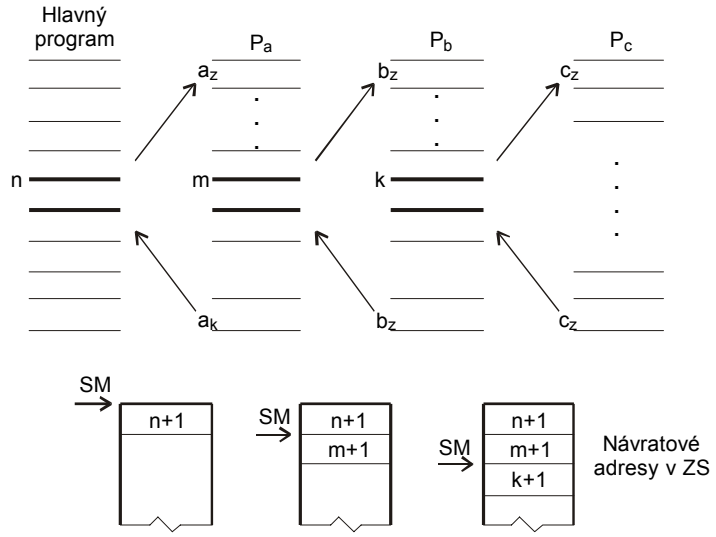
Inštrukcie programu	Obsah registrov ZS s adresami				
	0	1	2	3	4
Zapísať *A*	*A*	∗			
Zapísať *B*	*B*	*B*	∗		
Vynásobiť	*AB*	∗			
Zapísať *C*	*AB*	*C*	∗		
Zapísať *D*	*AB*	*C*	*D*	∗	
Spočítať	*AB*	*C* + *D*	∗		
Zapísať *E*	*AB*	*C* + *D*	*E*	∗	
Zapísať *F*	*AB*	*C* + *D*	*E*	*F*	∗
Vynásobiť	*AB*	*C* + *D*	*EF*	∗	
Zapísať *H*	*AB*	*C* + *D*	*EF*	*H*	∗
Spočítať	*AB*	*C* + *D*	*EF* + *H*	∗	
Deliť	*AB*	(*C* + *D*) / *EF* + *H*	∗		
Zapísať *K*	*AB*	(*C* + *D*) / *EF* + *H*	*K*	∗	
Odpočítať	*AB*	(*C* + *D*) / *EF* + *H* – *K*	∗		
Vynásobiť	*Y*	∗			
Zaznamenať	∗				

Ďalším využitím ZS je organizovanie prerušení a volania podprogramov, keď sa musia v priamom poradí do určenej hĺbky zapísať stavy prerušených programov a v opačnom poradí, na pokračovanie prerušených programov, tieto stavy obnoviť.

Aplikáciu ZS na pamätanie návratových adries pri odskoku do podprogramu v dôsledku jeho volania alebo prerušenia hlavného programu ilustruje Obr. 7.6. Na tomto obrázku označujú P_a, P_b a P_c obslužné programy prerušenia, ktorých začiatočné a koncové adresy sú postupne (a_z, a_k), (b_z, b_k), (c_z, c_k). Pri postupnom vnáraní do hlavného programu obslužných programov P_a, P_b do P_a a P_c, do P_b, sa príslušné návratové adresy $n + 1$, $m + 1$, $k + 1$ vkladajú do ZS s tým, že po skončení týchto programov sa v obrátenom poradí zo ZS volajú, čím sa zabezpečuje organizácia pokračovania v programe, ktorý bol prerušený.

Zásobníková pamäť (zásobník) môže byť organizovaná i tak, že súčasťou procesora je iba smerník a pamäťové bunky ZS vytvára vopred definovaný priestor v niektorej z úrovní pamäťového

podsystému počítača. Toto riešenie nie je náročné na spotrebu prvkov, avšak aktiváciou mimoprocesorovej úrovne pamäte pri realizácii zásobníkový inštrukcií sa znižuje celkový prístupový čas k ZS proti prípadu, keď je tento organizovaný ako registrová pamäť priamo v procesore. Pri ďalšej modifikácii organizácie ZS sa vychádza z nastaveného začiatočného stavu *SM* na jeho maximálneho hodnotu *M* (*SM* = *M*), v dôsledku čoho sa pri zápise smerník dekrementuje a pri čítaní inkrementuje.



Obr.7.6. Mechanizmus zápisu návratových adries do zásobníka.

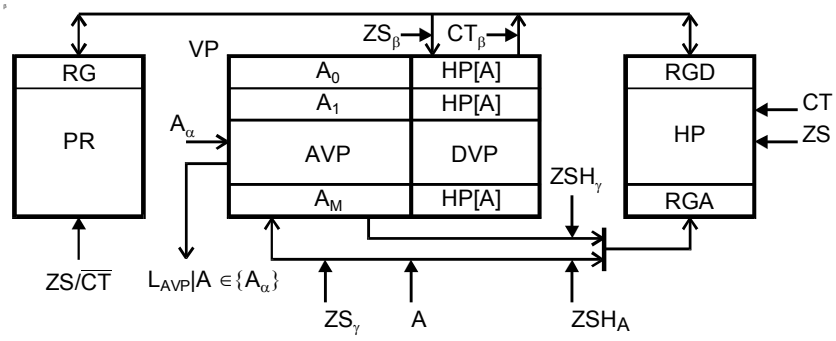
7.3 Vyrovnávacie pamäte typu cache

7.3.1 Štruktúrna organizácia vyrovnávacej pamäte

Vyrovnávacia pamäť (*VP*) je vo všeobecnosti určená na vyrovnávanie rýchlostí prenosu informácií medzi dvoma jednotkami číslicového počítača, v ktorých sa tieto spracúvajú s rozdielnymi rýchlosťami. VP je využívaná na prechodné uloženie dát alebo súboru dát [67], [84], [98], [161]. V prvom prípade je *vyrovnávacia pamäť* (*buffer*) reprezentovaná špecializovaným pamäťovým obvodom, ktorý môže byť využívaný tiež ako ukončovací ochranný obvod. Osobitný význam má VP, ktorá zvyšuje prenos súboru dát medzi procesorom a hlavnou pamäťou (HP). Prvý krát sa vyrovnávacia pamäť s uvedenou funkciou použila v počítači IBM 360/85 (r. 1968), kde tvorila prvú úroveň pamäťového podsystému, zatiaľ čo druhú a tretiu úroveň tvorili HP a sekundárna pamäť. Vytvárala v určitom zmysle „úkryt“ pri výmene informácií medzi procesorom a HP, čo ovplyvnilo jej názov – *pamäť typu cache* alebo skrátene – *cache*, ktorý sa v súčasnosti používa pre pamäťové jednotky s uvedenou funkciou.

Bez ujmy na všeobecnosti je v ďalšom uvedený princíp štruktúrnej organizácie a funkcie jednoúrovňovej koncepcie VP typu cache, ktorá vychádza z jej lokalizácie medzi procesorom a HP. Princíp tejto pamäte na úrovni štruktúrnej organizácie je na Obr. 7.7. Jej základné komponenty vo všeobecnosti sú:

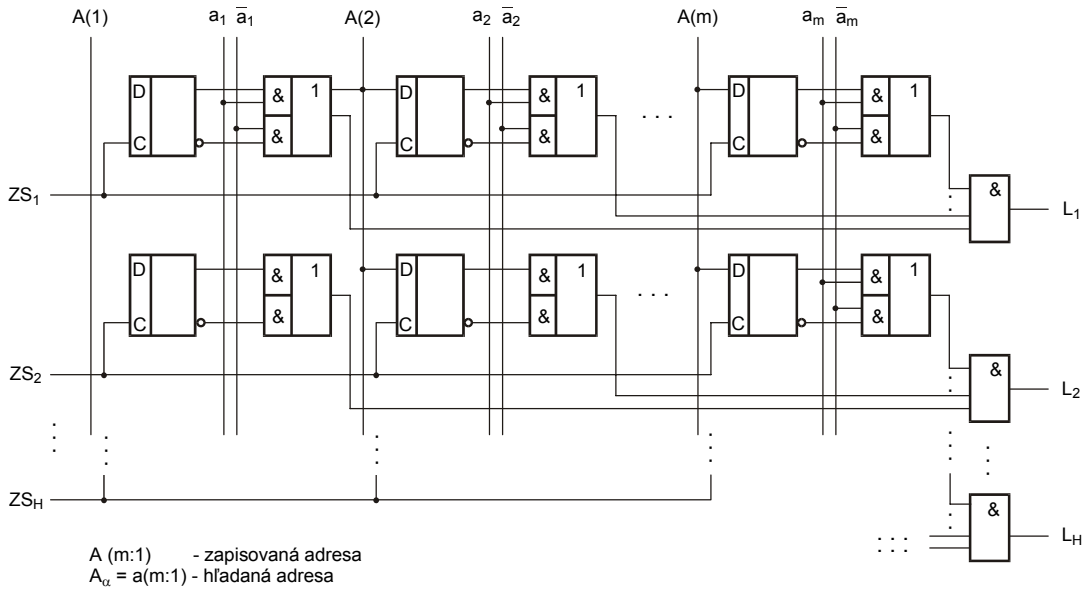
- *adresár VP* (*AVP*), v ktorom sú uložené adresy A_i kópií obsahov pamäťových buniek hlavnej pamäte (HP),
- *údajové pole VP* (*DVP*), v ktorom sú uložené kópie obsahov najčastejšie používaných pamäťových buniek hlavnej pamäti $HP[A_i]$, kde $i = 0,1, ..., n-1$.



Obr.7.7. Princíp štruktúrnej organizácie vyrovnávacej pamäte typu cache.

Pri aktivácii pamäte sa prostredníctvom prehľadávania adresára AVP najprv zisťuje, či adresované údaje nie sú uložené vo VP. Ak adresovaný údaj je vo VP, číta sa z nej bez prístupu do HP, čím sa viacnásobne zrýchli pamäťová operácia, pretože vybavovacia doba VP je rádovo 10 násobne kratšia ako vybavovacia doba HP. Keď adresovaný údaj vo VP sa nenachádza, číta sa z HP, avšak súčasne sa z neho vytvorí kópia do VP, takže pri opätovnej požiadavke na jeho sprístupnenie bude už k dispozícii vo VP. Súčasne sa s daným údajom presúvajú do VP aj údaje umiestnené v susedných pamäťových bunkách. Kópie údajov sa tak vytvárajú v dôsledku prenosu blokov údajov z HP do VP. Rozmery týchto blokov sú zväčša 8 až 128 B.

Pre zápis údajov z procesora do pamäte sa aktivuje vždy VP cache. Ak adresa zapisovaného údaja nie je vo VP, vyradí sa z nej nepotrebné slovo do HP, na miesto ktorého sa zapíše vysielaný údaj z procesora. Nepotrebné slová z VP do HP sa vyraďujú na základe rôznych stratégií (Kap. 7.5.3). Zápis údajov do HP sa vykonáva iba v rámci procesu vyraďovania nepotrebných slov z VP, z čoho vyplýva požiadavka na takú kapacitu VP a veľkosť prenášaných blokov údajov, aby komunikácia medzi VP a HP bola minimálna.



Obr.7.8. Schéma asociatívnej pamäte.

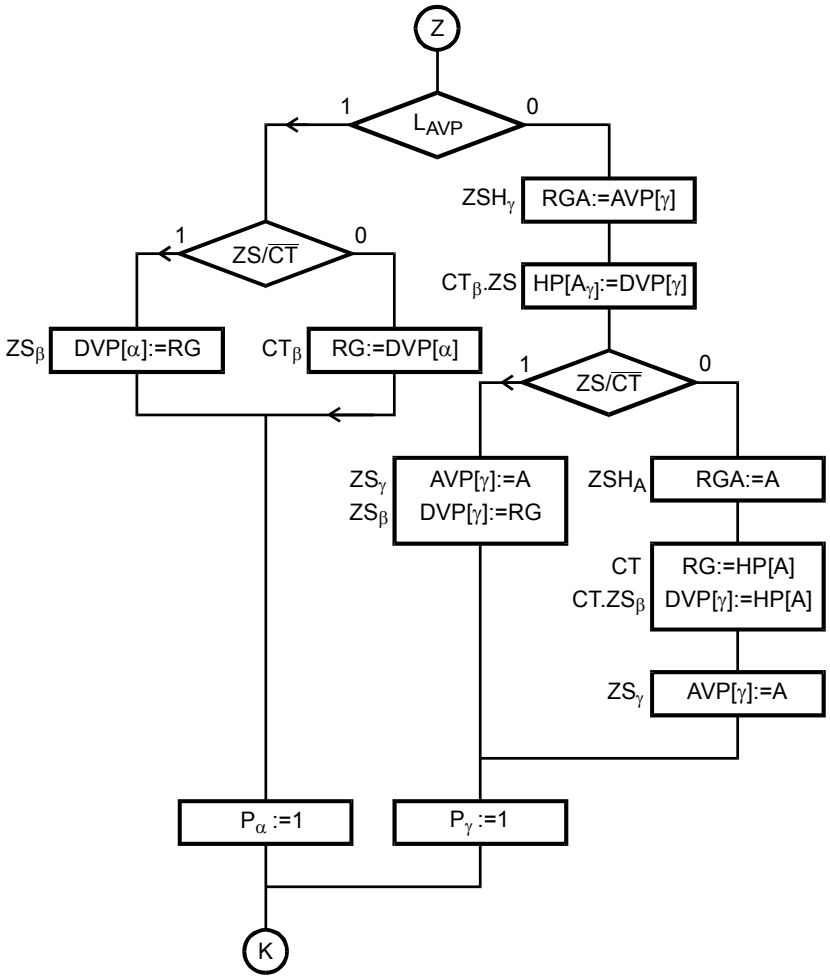
Hľadanie adresy v AVP sa uskutočňuje na princípe prehľadávania pomocou obsahu, v dôsledku čoho táto časť VP je realizovaná ako asociatívna pamäť. Schéma zapojenia asociatívnej pamäte je

uvedená na Obr. 7.8, v ktorej bit má k dispozícii *logický obvod ekvivalencie* a každé *m*-bitové slovo je vybavené *m*-vstupovým súčinovým obvodom. Tieto logické obvody predstavujú *riadkový komparátor*, prostredníctvom ktorého sa vyhľadá riadok *a* (*a* = 0, 1, ..., *M*), v ktorom dochádza ku zhode medzi hľadanou adresou *A* a zapísanou adresou A_α kópie údaja $HP[A_\alpha]$ vo VP. Adresa A_α sa do AVP zapisuje mikrooperáciou ZS_γ) $AVP\,[\gamma] := A_\alpha$, kde index γ s významom čísla riadku vo VP, určujú riadiace obvody mechanizmu vyraďovania nepotrebných slov z VP. Zhoda hľadaných a ich zapísaných kópií v ľubovolnom riadku asociatívnej VP je indikovaná prostredníctvom stavovo-informačnej premennej

$$L_\alpha \mid A \in \{A_\alpha\}$$

kde

$$L_{AVP} = \bigvee_{\alpha=0}^{M} L_{á}$$



Obr.7.9. Algoritmus riadenia vyrovnávacej pamäte.

7.3.2 Princíp riadenia vyrovnávacej pamäte

Algoritmus riadenia prenosu údajov medzi procesorom, VP a HP pri zápise a čítaní je uvedený na Obr. 7.9. Po predchádzajúcom potvrdení (L_{AVP} = 1) nájdenia adresy operanda sa ďalší prístup do

VP realizuje pomocou signálu $ZS/\overline{CT} = 1$ pri zápise a $ZS/\overline{CT} = 0$ pri čítaní, zatiaľ čo riadenie HP sa uskutočňuje osobitnými signálmi *ZS* pri zápise a *CT* pri čítaní. Skončenie aktivácie HP sa indikuje pomocou logickej premennej L_{HP} . Riadok AVP, určený na vyradenie, je označený indexom γ. Po aktivácii α. alebo γ. riadku VP sa nastaví príslušná premenná P_α, resp. P_γ s významom príznaku, ktorý sa využíva pri stratégii vyraďovania nepotrebných údajov z VP. V dôsledku uvedeného sa zápis do VP ako mikrooperácia v tvare:

$$ZS/\overline{CT}.ZS_{\hat{a}})\, DVP[\hat{a}] := RG$$

a čítanie z VP ako mikrooperácia

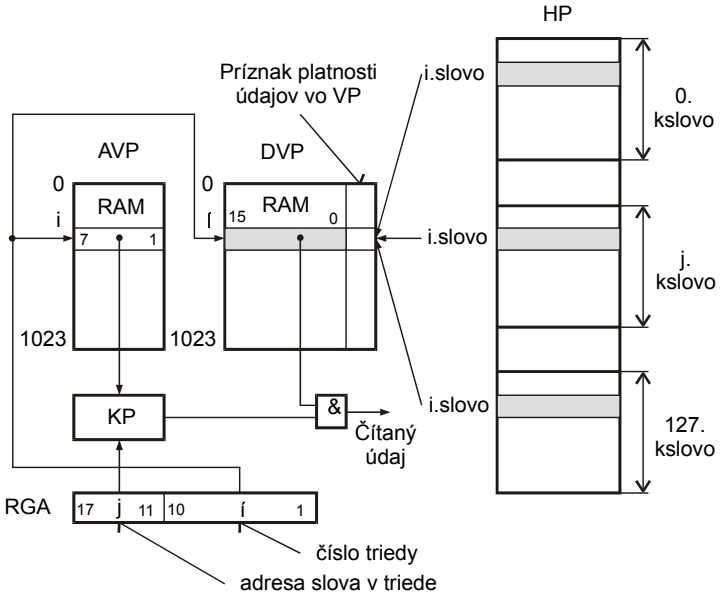
$$\overline{ZS/\overline{CT}}.CT_{\hat{a}})RG := DVP[\hat{a}]$$

kde $\beta = \gamma \vee \alpha$.. HP sa aktivuje zloženými riadiacimi signálmi, ktoré inicializujú mikrooperácie v tvare:

$$CT_{\hat{a}}.ZS)\, HP[A_\gamma] := DVP[\gamma]$$

resp.

$$ZS_{\hat{a}}.CT)\, DVP[\gamma] := HP[A]$$



Obr.7.10. Adresovo-asociatívny prístup do VP organizovanej na báze pamäte RAM.

keď predtým sa adresa sprístupňovaných údajov v HP nastavuje mikrooperáciou

$$ZSH_{\tilde{a}})\, RGA := AVP[\tilde{a}]$$

pri zápise vyradených slov z VP

$$ZSH_A)\, RGA := A$$

pri tvorbe kópií slov vo VP ich čítaním z HP.

Adresový princíp sprístupňovania informácií uplatňovaný v štandardných pamätiach RAM neumožňuje ich priamo využiť na realizáciu VP s asociatívnym výberom. Tieto VP, ktorých hlavné

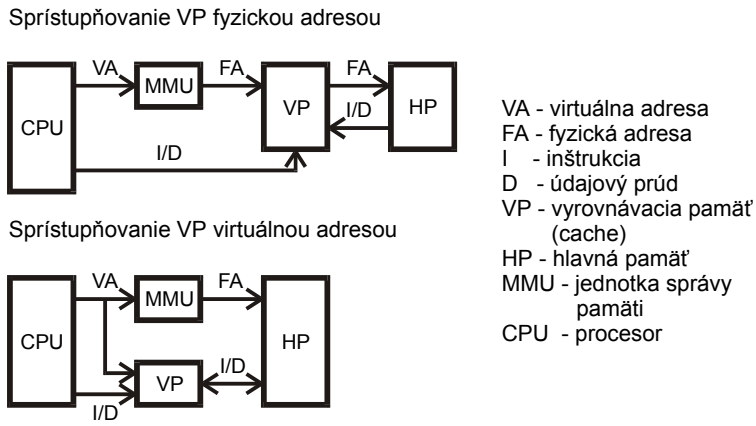
komponenty realizované v ich východiskovej verzii na Obr. 7.7 ako jeden integrovaný obvod, sú *adresár* (*asociatívna pamäť AVP*) a *vlastná vyrovnávaciu pamäť* (*údajové pole DVP*). Príslušné obvody sú však cenovo pomerne náročné a ich kapacita je obmedzená. Už pri kapacite HP väčšej, ako 64 kB, plne asociatívny výber sa stáva pomalým, pretože asociatívne prehľadávanie sa uskutočňuje podľa pomerne dlhého kľúča (adresy), z čoho vyplývajú uvedené nedostatky takto riešenej pamäte. Východiskom je návrh pamäte, ktorá má *obmedzený stupeň asociativity*. V takejto pamäti sa uplatňuje tzv. *adresovo-asociatívny výber*, ktorý umožňuje na jej realizáciu využiť štandardné pamäte typu RAM. Podstatu tohto prístupu ilustruje Obr. 7.10 (riešenie použité v niektorých minipočítačoch fy DEC), z ktorého vyplýva, že priestor HP je rozdelený do 1024 tried. Každá trieda sa skladá zo 128 slov rovnomerne rozložených v HP s odstupom 128 slov, z ktorých vždy iba jedno môže byť zobrazené v príslušnom riadku VP, tzv. segmente. V *i*. segmente VP je takto zobrazená kópia (adresa pamäťovej bunky a jej obsah) vždy iba jedeného (*j*.) zo slov *i*. triedy. Pri výbere kópie z VP sa najprv podľa najnižších desiatich bitov adresy A vyberie adresovým prístupom obsah *i*. riadku, t.j adresa, ktorá sa v komparátore KM porovná s najvyššími 7 bitmi adresy A. Zhoda porovnávaných údajov indikuje, že hľadaný údaj sa vo VP nachádza a preto sa z príslušného riadku DVP môže čítať. V opačnom prípade nezhoda porovnávaných údajov indikuje, že vyžadovaný údaj sa vo VP nenachádza. Osobitným jednobitovým príznakom v každom riadku DVP sa indikuje platnosť údajov, čo sa využíva najmä pri stratégii vyraďovania nepotrebných slov, resp. blokov slov z VP.

Adresovo-asociatívny prístup do VP sa môže organizovať tak, že z *i*. triedy je vo VP umiestnených *k* slov, tzn. že VP sa skladá zo skupín *k* segmentov, z ktorých sa zobrazuje *k* kópií adresovanej triedy slov. VP ktorá využíva takýto prístup sa nazýva viaccestná asociatívna pamäť. Touto problematikou sa podrobnejšie zaoberá Kap. 7.3.3 zameraná na problematiku mapovania súčasných VP typu cache.

7.3.3 Mapovanie vyrovnávacej pamäte cache

Sprístupňovanie cache sa uskutočňuje prostredníctvom:

- fyzickej adresy (VAX 8600, Intel i486, MIPS R 3000),
- virtuálnej adresy (Intel i 860),

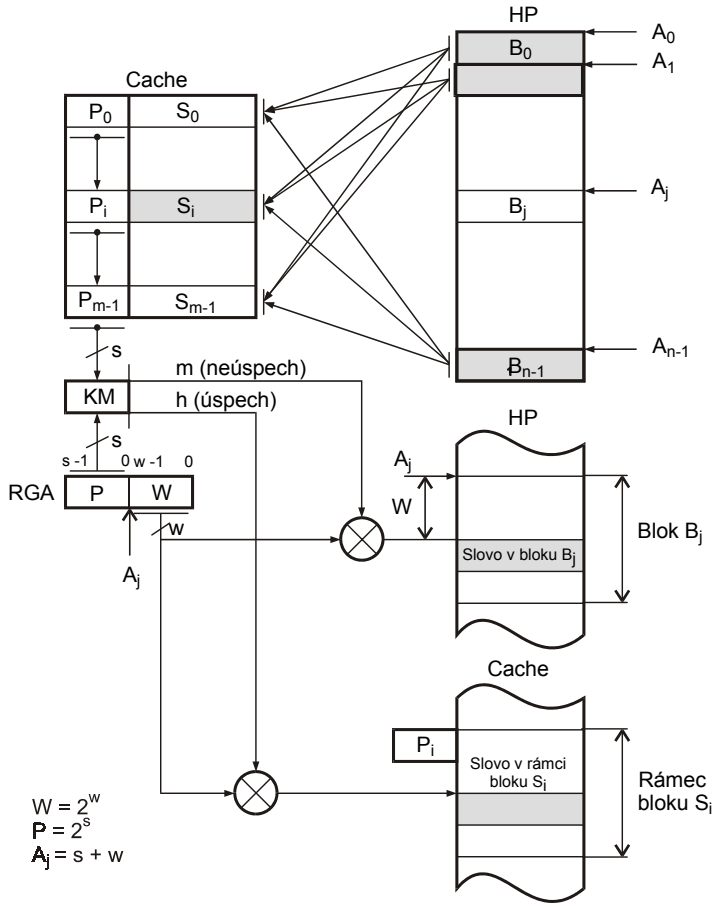


Obr.7.11. Modely adresovania vyrovnávacej pamäte.

ktorým sú definované dva základná modely adresovania cache, uvedené na Obr. 7.11. Model fyzického adresovania vychádza z prehľadávania cache (*fyzicky adresovaná cache*), ktoré sa uskutočňuje po preklade virtuálnej adresy v *jednotke správy pamäte* (*MMU – M*emory *M*anagement *U*nit) alebo v *jednotke preložených adries* (*TLB – T*ranslation *L*ookaside *B*uffer). Ak adresované inštrukcie alebo údaje sú v cache nájdené, jej riadiace obvody potvrdzujú túto skutočnosť

generovaním signálu *úspešného hľadania* (*h – h*it), v opačnom prípade cache generuje signál *neúspešného hľadania* (*m – m*iss). Po neúspešnom prístupe do cache sú vyslané kópie hľadaných údajov z pamäte vyššej úrovne, kde sú uložené ich originály. Vysiela sa vždy celý údajový blok alebo riadok vopred definovanej veľkosti, v ktorom sa adresovaná položka nachádza. Výhodou adresovania cache priamo fyzickou adresou je, že nevzniká potreba jej uvoľňovania pri neúspešnom hľadaní vyžadovaných položiek.

Pri virtuálnom adresovaní cache je táto adresa paralelne preložená v MMU. Fyzická adresa generovaná v MMU je uložená s cieľom jej spätného zápisu pri obnove údajov v cache. Virtuálne adresovanie cache zvyšuje účinnosť rýchlejšieho prístupu do cache, vďaka prekrývaniu s prekladom adresy v MMU.



Obr.7.12. Štruktúrna organizácia plne asociatívnej pamäte cache.

Prenos údajov medzi HP (pamäťou vyššej úrovne) a cache sa uskutočňuje ako prenos skupiny slov, ktorú reprezentuje blok alebo riadok definovaného rozmeru. Na odlíšenie skupiny slov sa táto skupina v HP nazýva *blok* a v cache, *rám bloku*. Organizácia prenosu údajov medzi dvoma pamäťovými jednotkami sa vo všeobecnosti nazýva *mapovanie pamäte*. Konečná výkonnosť toho ktorého spôsobu mapovania závisí od prístupovej stratégie adresovania cache, organizácie cache a používaného riadenia. V Kap.7.3.2 boli uvedené dve základné koncepcie riešenia VP: s *asociatívnym* a *adresovo-asociatívnym prístupom*. V ďalšom sú uvedené niektoré spôsoby mapovania pamäte, podľa ktorých sa rozlišujú *asociatívne pamäte cache*, cache *s priamym adresovaním*, cache s *obmedzeným stupňom asociativity k* = 1, 2, ... a cache so *sektorovou organizáciou*.

Plne asociatívna pamäť cache. Štruktúrna organizácia plne asociatívnej pamäte cache je na Obr. 7.12. Nech

A_j je bázová adresa bloku B_j s veľkosťou k slov v HP, kde $j = 0, 1, ..., n\text{-}1$,
P_i - kópia bázovej adresy bloku B_j v cache, t.j. príznak (kľúč) asociatívneho výberu,
S_i - rám bloku v cache, kde $i = 0, 1, ..., m\text{-}1$,
pričom

$$P_j \in \{ A_0, A_1, ..., A_j, ..., A_{n-1}\}$$

Pre $n >> m$ je mapovanie HP do cache definované zobrazením:

$$map_j : B_j \rightarrow S_i \ , \ i = 0, 1, ..., m-1, j = 0, 1, ..., n-1$$

tzn. že ľubovolný blok B_j môže mať svoju kópiu (rám bloku S_i) v ľubovolnej pozícii cache, definovanej príznakom (kľúčom asociatívneho výberu) P_i . Pre dĺžku *adresy príznaku* (kľúča) P rámu bloku S, *posuvu slova* W v ráme bloku S (bloku B) a *adresy* A slova (bajtu) v bloku B hlavnej pamäte platí:

$$P = 2^s \ , \ W = 2^w \ , \ A = s + w$$

kde
s jepočet bitov príznaku P,
w - počet bitov relatívneho posuvu (offsetu) slova W v ráme bloku S (bloku B).

Napríklad, ak HP s kapacitou $C = 1$ *GB* (Gslov) reprezentuje $n = 2^{27} = 128$ *M* blokov, z ktorých každý obsahuje $W = 2^5 = 32$ slov (bajtov), potom $w = 5$ bitov, $s = 27$ bitov a dĺžka adresy A slova v HP je 5 + 27 = 32 bitov, pretože $C = n \times W = 2^{27+5} = 1$ *GB*.

Úspešný a neúspešný prístup do cache indikujú stavové premenné h (hit) a m (miss) nasledovne:

$$h \mid A_j = P_0 \vee ... \vee P_i \vee ... \vee P_{m-1}$$
$$m \mid A_j = \overline{P}_0 \wedge ... \wedge \overline{P}_i \wedge ... \wedge \overline{P}_{m-1}$$

Pamäť cache s obmedzeným stupňom asociativity. Prístup do asociatívnej VP cache prostredníctvom VA je príliš nákladný, pretože *VA >> FA*, v dôsledku čoho príznakový kľúč asociatívneho výberu má veľký rozmer. Efektívne riešenie poskytuje koncepcia VP s adresovo - asociatívnym prístupom (Obr. 7.10), ktorá umožňuje obmedziť stupeň asociativity definovaný koeficientom k.

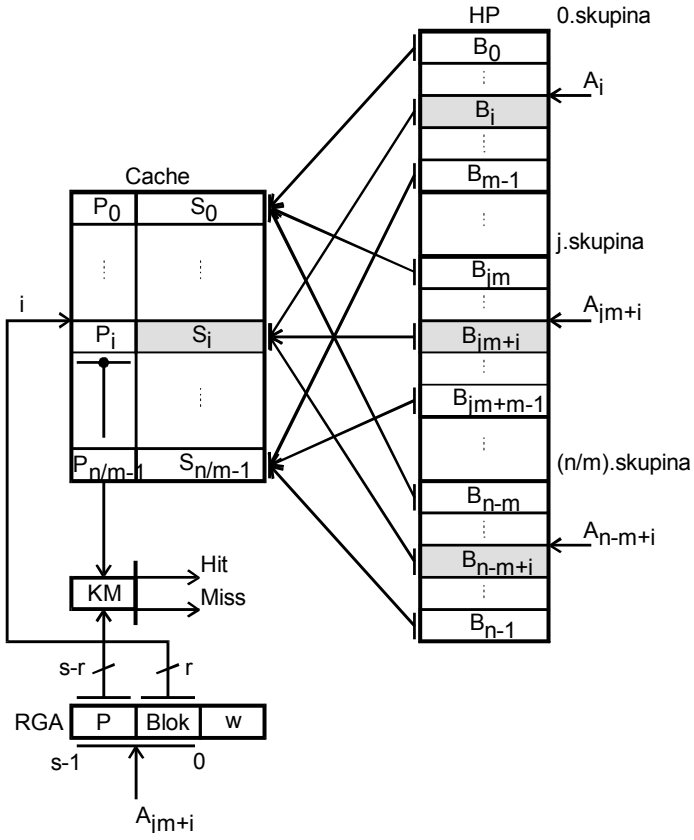
Pamäť cache so *stupňom asociativity* $k = 1$ (VP s priamym mapovaním) predstavuje riešenie, pri ktorom je mapovanie definované zobrazením (Obr.7.13):

$$map_j : B_j \rightarrow S_i$$
$$j = 0, 1, ... , n/m\text{-}1, \ i = j \ (mod \ m)$$

kde n / m je počet skupín blokov, na ktoré je HP rozdelená.

V tomto prípade iba jeden blok z každej skupiny blokov HP môže byť mapovaný do jedného a toho istého rámu bloku v cache. Pretože bloky B_j hlavnej pamäte sa pri $k = 1$ mapujú vždy iba do *jedného rámu bloku* S_i v cache nazývajú sa tieto pamäte *jednocestné asociatívne cache*, na rozdiel od *viaccestných* (k-cestných), v ktorých definované bloky B_j hlavnej pamäte sa mapujú do *viacerých*

rámov blokov (do *k* rámcov blokov S_{ik} , S_{ik+1} , ..., S_{ik+k-1}) v cache (Obr.7.14). To znamená, že ak bloky B_0, B_1, ... , B_{n-1} v HP sú usporiadané do *n*/*m* skupín s veľkosťou *m* blokov, potom *i*-te bloky z ľubovoľnej skupiny 0,1, ... , *n*/*m* môžu byť mapované do *i*-tej položky VP. Táto položka sa identifikuje na základe adresového výberu, pričom jej zložkami sú príznak *skupiny blokov* P_i a *j*-ty rám bloku S_j, ktorý zodpovedá kópii *bloku* B_{jm+i} vo VP, pretože platí:



Obr.7.13. Štruktúrna organizácia pamäte cache so stupňom asociativity $k = 1$.

$$P_i \in \{A_i, A_{i+1}, \ldots, A_{jm+i}, \ldots, A_{n-m+i}\}$$

Formát VA, ktorá je reprezentovaná adresou A_{jm+i} a posuvom slova *w* v príslušnom bloku je:

$$VA = P((s - r - 1) : 0) \,.\, Blok((r - 1) : 0) \,.\, w$$

kde

s je počet bitov na zobrazenie čísla bloku,

r - počet bitov na zobrazenie počtu blokov v skupine ($r = log_2 m$).

Skupinovo asociatívna cache. Je osobitný prípad cache s obmedzeným stupňom asociativity pre $k \neq 1$ (cache s *k* - stupňovou asociativitou, resp. *k*-cestná asociatívna cache). Schéma mapovania v *k*-cestnej asociatívnej cache je uvedená na Obr. 7.15.

Pre rámce blokov P_{ik} , P_{ik+1}, ..., $P_{ik+k-1} \in \{ A_i, A_{i+1}, \ldots, A_{iq+i}, \ldots, A_{n-q+i}\}$ je mapovanie HP do cache definované zobrazením:

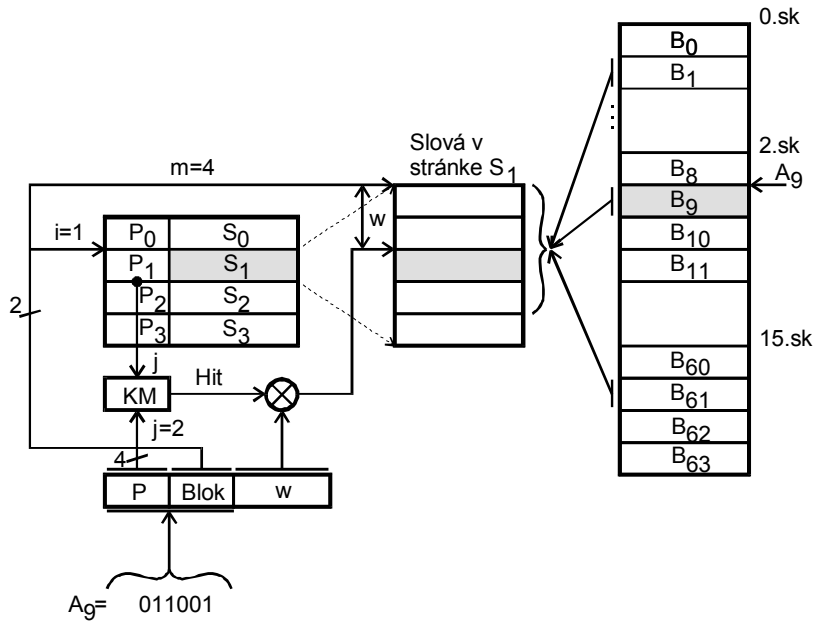
$$map_j : B_j \rightarrow S_{ik} , S_{ik+1}, \ldots, S_{ik+k-1}$$

kde pre $\alpha \in \{ik, ik + 1, \ldots, ik + k - 1\}$ platí:
$\alpha = j \ (mod \ q)$, $j = 0, 1, ..., n / q – 1$

Ak $k = m$, potom $q = m / k = 1$.

Príklad 7.1. *Ilustrujte mapovanie vo VP v prípade, ak n = 64 a m = 4. Riešenie je uvedené na Obr. 7.14*

.

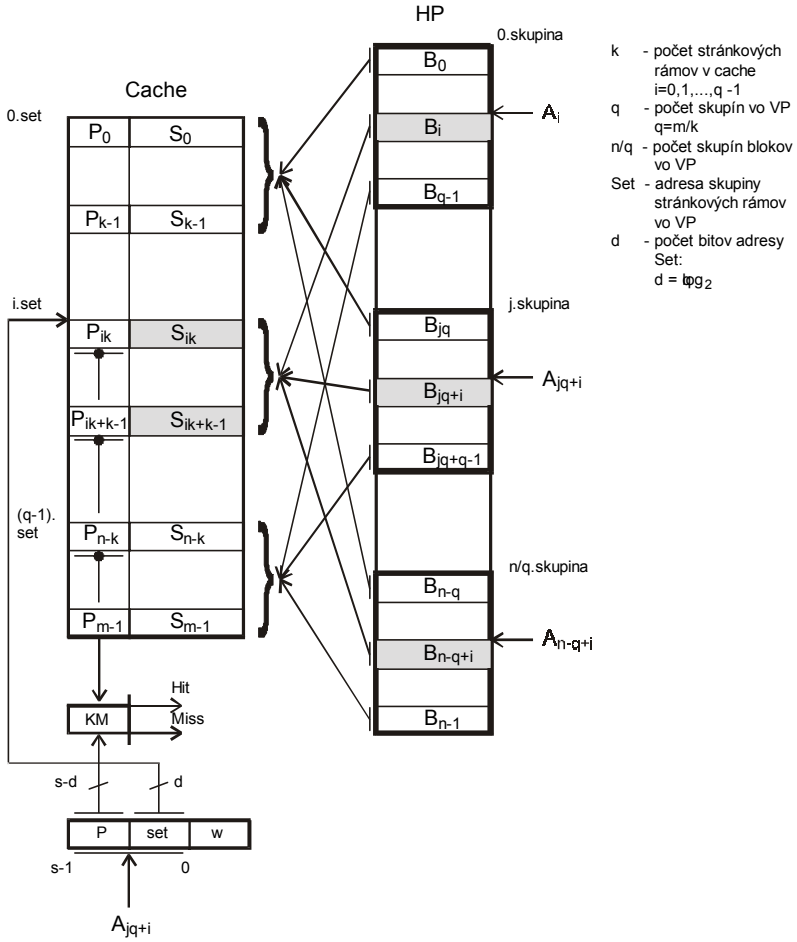


Obr.7.14. Príklad mapovania cache so stupňom asociativity k = 1 (jednocestnej asociatívnej cache).

•

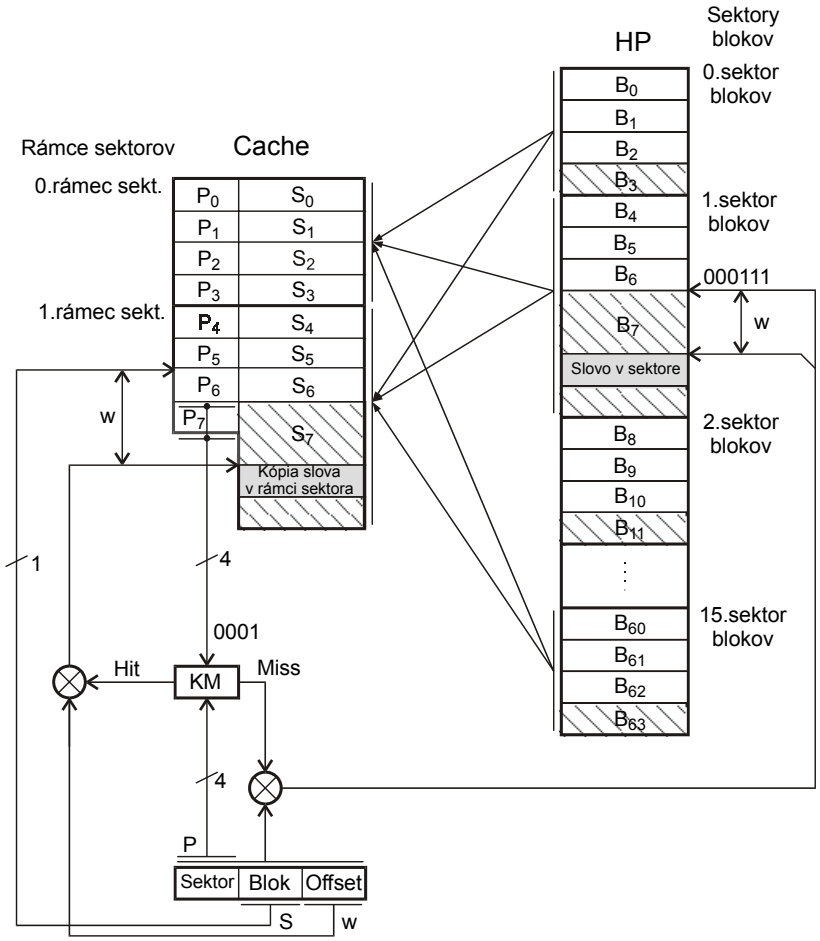
Pamäť cache so sektorovou organizáciou. Umiestnenie blokov v tomto prípade predstavuje zovšeobecnenie predchádzajúcich schém sprístupňovania cache. Táto koncepcia vychádza z predstavy rozdelenia cache a HP do *sektorov* s vopred definovaným rozmerom, ktoré môžu byť umiestnené v ľubovolnom ráme sektora v cache. Princíp sektorovej organizácie cache je uvedený na Obr. 7.16.

Požiadavky na aktiváciu pamäte sú formulované na sprístupňovanie blokov a nie sektorov. Tento prístup môže byť eliminovaný prostredníctvom vyhľadávania príznaku sektora na základe plne asociatívneho prístupu. Ak vyhľadanie rámu sektora je úspešné, je úspešné i vyhľadanie bloku v ráme sektora. Pri takejto organizácii vyhľadávania blokov je dôležitá určiť, či blok v danom sektore je platný alebo platný nie je. Za týmto účelom každý rám bloku má pridaný *bit platnosti bloku* v sprístupňovanom ráme sektora, ktorý indikuje, či *blok je platný* alebo *neplatný*. Na Obr. 7.16 je uvedený príklad štruktúry HP rozdelenej na 16 sektorov, každý veľkosťou 4 bloky. Pamäť cache pozostáva z dvoch rámov sektorov, do ktorých sa môže mapovať ľubovolný sektor HP so svojimi blokmi. V ilustrovanom príklade je znázornené mapovanie do cache kópie slova s posuvom (offsetom) *w* v bloku 7 lokalizovanom v 1. sektore HP.



Obr.7.15. Schéma mapovania v *k*-stupňovej asociatívnej pamäti.

V porovnaní s plne asociatívnou cache alebo s viaccestnou asociatívnou cache prednosť sektorového mapovania pamäte cache spočíva v implementovaní algoritmu rozličného navracania blokov a v ekonomickom vykonávaní plne asociatívneho vyhľadávania pomocou ohraničeného počtu príznakov rámcov sektorov.



Obr.7.16. Pamäť cache so sektorovou organizáciou.

7.4 Hlavná pamäť

Hlavná pamäť (*HP*) tvorí v hierarchicky usporiadanom systéme ČP pamäť druhej úrovne, medzi ktorou a procesorom sa vkladá podsystém vyrovnávacích pamätí typu cache [67], [98], [121], [161], [162]. Pôvodne sa HP nazývala operačná pamäť najmä preto, že sa priamo zúčastňovala na vykonávaní aritmetických a logických operácií procesorom. V súčasnosti túto úlohu preberá úroveň vyrovnávacích pamätí typu cache a preto uvedený termín stratil svoj pôvodný význam. Hlavná pamäť je určená na zápis inštrukcií a údajov vykonávaného programu, ktorého aktuálne časti sa kopírujú do pamätí cache, takže pri realizácii inštrukcií týchto aktuálnych častí programu procesor komunikuje s pamäťou cache. Efektívne a rýchle vykonávanie základnej funkcie HP ovplyvňuje najmä jej štruktúrna organizácia a organizácia jej sprístupňovania z jednotlivých hierarchických úrovní pamäťového podsystému.

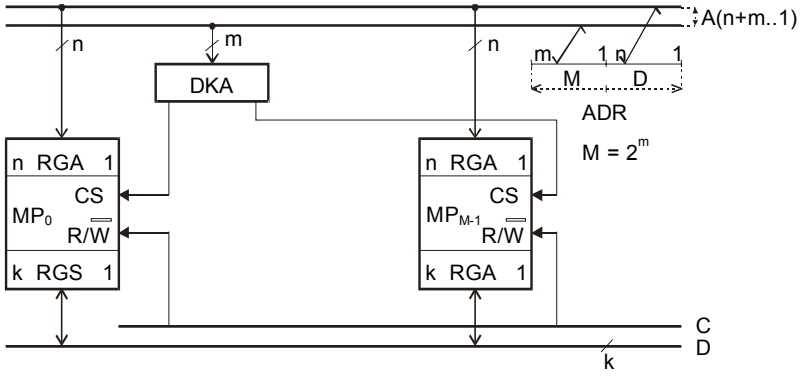
7.4.1 Modulárna organizácia pamäte

Z výrobno-technologických dôvodov je HP reprezentovaná z *pamäťových modulov*, ktorých kapacita je v súčasnosti rádovo 32 – 64 MB. V najjednoduchšom prípade sa HP skladá z jedného pamäťového modulu [161]. Modulová organizácia HP má rad výhod, spomedzi ktorých medzi

najvýznamnejšie patria:

1. Rozdelenie HP na menšie celky (moduly) umožňuje znížiť čas cyklu pamäte (vybavovaciu dobu), ktorá závisí od jej kapacity.
2. Umožňuje prekrývať fázy pamäťového cyklu, pri aktivácii pamäte.
3. Podľa požiadaviek používateľa podporuje rozšíriteľnosť kapacity pamäte.

Adresovanie pamäťových buniek v modulárne organizovanej *hlavnej pamäti* (*HP*) sa uskutočňuje prostredníctvom adresy (*ADR*), ktorá sa skladá z dvoch segmentov *M* a *D* (Obr.7.17).

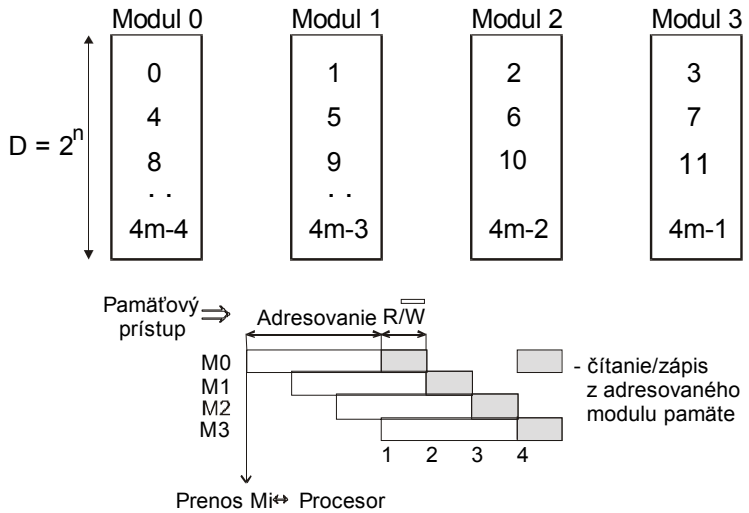


Obr.7.17. Modulárna organizácia pamäte.

Pri sprístupňovaní pamäte jej modulárna organizácia umožňuje prekrývať adresovanie nasledujúceho modulu s vlastným procesom čítania/zápisu z/do predchádzajúceho modulu (Obr. 7.18). Vo všeobecnosti existujú tri prístupy organizovania modulárnej pamäti, ktoré vyplývajú z nasledujúcej interpretácie adries pamäťových buniek:

$ADR(n + m..1) = M\,(m..1)\,.\,D\,(n..1)$
$ADR(n + m..1) = D\,(n..1)\,.\,M\,(m..1)$
$ADR(n + m..1) = M_1\,(m_1..1)\,.\,D\,(n..1)\,.\,M_2\,(m_1..1)$

kde $m_1 + m_2 = m$.

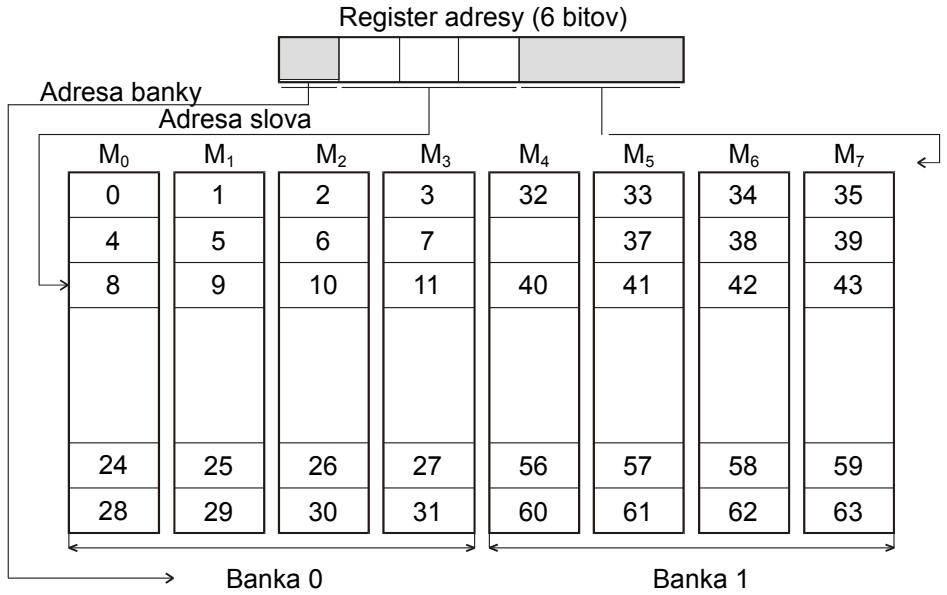


Obr.7.18. Prekrývanie prístupu do pamäte s modulárnou štruktúrou.

Uvedené alternatívy sprístupňovania HP reprezentujú:

- *lineárny prístup* k súvislej oblasti HP v rámci modulov,
- prekrývaný prístup k súvislej oblasti HP distribuovanej medzi jednotlivými modulmi,
- *prekrývaný prístup* k súvislej oblasti HP distribuovanej medzi jednotlivými modulmi organizovanými *v skupinách* (v pamäťových bankách), čím sa zvyšuje odolnosť proti poruchám v moduloch, v dôsledku izolovanej činnosti jednotlivých pamäťových bánk.

Na Obr. 7.19 je uvedený príklad adresovania dvojbankovej HP. V každej z dvoch bánk sa uskutočňuje štvorcestné prekrývanie sprístupňovania pamäťových modulov. Činnosť jednotlivých bánk je vzájomne nezávislá, čo sa využíva na zvýšenie odolnosti pamäte voči poruchám.



Obr.7.19. Organizácia štvorcestného prekrývania v dvojbankovej HP.

7.4.2 Riadenie prístupu do pamäte

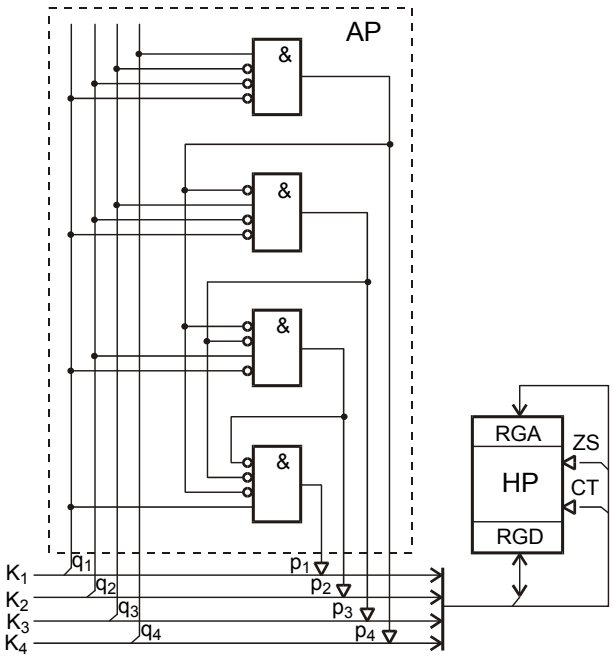
Pamäťový priestor HP sa využíva viacerými zariadeniami ČP, s ktorými HP pri vykonávaní programu komunikuje po systémovej zbernici (procesor, sekundárne pamäte), alebo priamo (pamäte cache) prostredníctvom svojich riadiacich jednotiek. Tieto zariadenia pracujú súčasne a nezávisle jeden od druhého, pričom generujú svoje požiadavky na aktiváciu HP s cieľom zápisu alebo čítania inštrukcií a údajov používateľských a systémových programov. Pretože v každom časovom okamihu môže HP komunikovať iba s jedným z uvedených zariadení, do popredia vystupuje problém prideľovania HP viacerým zariadeniam. Prideľovanie HP sa môže uskutočňovať multiplexovaním zbernice alebo organizáciou viackanálového prístupu. Viackanálový prístup do HP reprezentuje všeobecný princíp prideľovania zbernice viacerým zariadeniam. Základným predpokladom jeho organizácie je, že požiadavky na pridelenie HP sa obsluhujú v takom poradí, v akom sú generované. Pri konfliktnej situácii, keď vzniká niekoľko požiadaviek súčasne, uprednostní sa tá požiadavka, ktorá má *vyššiu prioritu*. Nech signály q_1, q_2, ..., q_M označujú požiadavky na pridelenie HP od *M* zariadení, pričom ich priorita sa definuje poradovým číslom tak, že menšie hodnoty čísla zodpovedajú požiadavke zariadenia s vyššou prioritou a signály p_1, p_2, ..., p_M povolenie príslušným kanálom prístup do HP. Hodnotou premennej $p_k = 1$, je definované pridelenie *k*. kanálu hlavnej pamäti v prípade, ak sa generuje požiadavka $q_k = 1$ nie sú generované

požiadavky $q_1, ..., q_{k-1}$ z kanálov s vyššou prioritou a nie je začatá obsluha niektorého z kanálov $k = 1, ..., M$, t.j.:

$$p_k = \overline{q}_1 \cdots \overline{q}_{k-1} q_k \overline{p}_{k+1} \cdots \overline{p}_M \tag{7.1}$$

V prípade štyroch prístupových kanálov (napríklad procesor, disková pamäť a dve vstupno/výstupné zariadenia) k HP zo vzťahu (7.1) vyplýva:

$$p_1 = q_1 \overline{p}_2 \overline{p}_3 \overline{p}_4 \ , \quad p_2 = \overline{q}_1 q_2 \overline{p}_3 \overline{p}_4 \ , \quad p_3 = \overline{q}_1 \overline{q}_2 q_3 \overline{p}_4 \ , \quad p_4 = \overline{q}_1 \overline{q}_2 \overline{q}_3 p_4$$



Obr.7.20. Organizácia viackanálového prístupu do HP.

Princíp pamäte s viackanálovým prístupom je zobrazený na Obr. 7.20. Kanály generujú požiadavky $q_k = ZS_k \vee CT_k$, ktoré nadobúdajú logickú hodnotu 1 v okamihu aktivácie HP na zápis alebo čítanie informačných slov. Tieto signály sa spracujú v *arbitri priorít* (*A*P), ktorého obvodové riešenie vychádza z realizácie vzťahu (7.1).

Od organizácie viackanálového prístupu do HP sa odlišuje organizácia viacprístupovej pamäte, ktorá umožňuje rôzne kombinácie súčasného prístupu (súčasný zápis z viacerých zdrojov, súčasný zápis a čítanie a pod.).

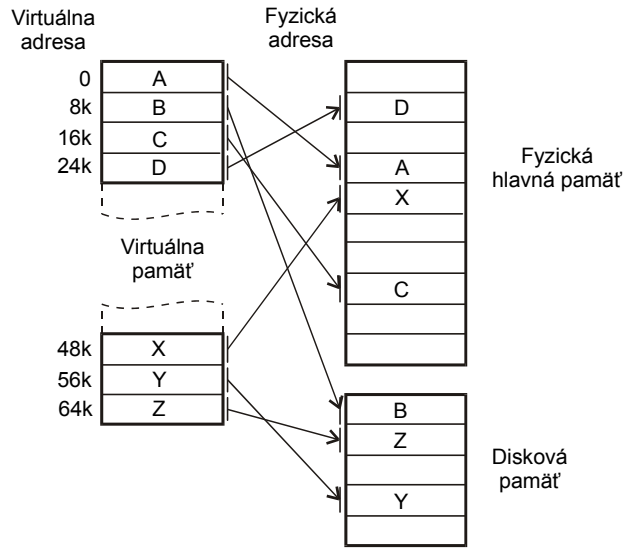
7.5 Systém virtuálnej pamäte

Pri spracúvaní informácií v ČP je typická taká situácia, pri ktorej iba časť programu, pozostávajúceho z kódu inštrukcií a údajov je uložená v HP, zatiaľ čo ostatná časť programu je uložená v diskovej (sekundárnej) pamäti (DP). V tomto prípade sa stáva zásadnou otázka zabezpečenia výmeny informácií medzi HP a DP. Riešenie tohto problému osobitne vystupuje do popredia pri spracúvaní rozsiahlych úloh a pri prideľovaní pamäťového priestoru viacerým

používateľom, keď principiálne nemožno do HP s obmedzenou kapacitou umiestniť celý program, ktorý sa má vykonať v ČP. Okrem toho, v každom okamihu chodu počítača sa súčasne spracúva viac procesov, z ktorých každý má vyhradený svoj adresový priestor. Takáto situácia sa výrazne prejavuje v multiprogramových systémoch, v ktorých má každý používateľ k dispozícii svoj vlastný adresový priestor, definovaný identifikátorom príslušného procesu. Bolo by príliš neefektívne a cenovo náročné, aby každému procesu bol v HP venovaný celý jeho adresový priestor, keď väčšina procesov vo vopred definovanom časovom intervale využíva iba malú časť tohto priestoru. Preto ČP má k dispozícii prostriedky na pridelenie jednotlivým procesorom menšie časti adresového priestoru HP, ktorá v tomto prípade vykonáva funkciu *fyzickej pamäte* (*FP*) pamäťového podsystému. Tieto prostriedky reprezentuje *virtuálna pamäť* (*VP*) s fiktívnym adresovým priestorom potrebným na vykonanie všetkých programov a prebiehajúcich procesov [21], [67|, [98]. Koncepcia VP spočíva v rozdelení priestoru FP na bloky slov a ich prideľovaní jednotlivým procesom na základe transformácie virtuálnych adries priestoru VP. Organizácia VP musí zabezpečiť ochranu procesov pred komunikáciou s blokmi, ktoré danému procesu nie sú pridelené.

7.5.1 Organizácia virtuálnej pamäte

V architektúrach ČP sa uplatňuje rad prístupov na realizáciu virtuálnej pamäte VP. V súčasnosti je najčastejšie využívaná koncepcia virtuálnej pamäte, ktorá riadi a organizuje činnosť dvoch úrovní pamätí, ktorými sú HP a DP. Bez ujmy na všeobecnosti je v ďalšom uvedený princíp organizácie VP ako jednoúrovňovej, ktorej kapacita sa rovná súčtu kapacít HP a DP. Na Obr. 7.21 je uvedený príklad *transformácie adries* VP na adresy FP pre programy, z ktorých jeden pozostáva zo štyroch blokov slov (A, B, C, D) a druhý z troch blokov (X, Y, Z). Bloky A, B a C sú fyzicky lokalizované v HP a blok D je fyzicky lokalizovaný v DP. Pre vykonanie druhého programu blok X je fyzicky lokalizovaný v HP a blok Y a Z sú fyzicky lokalizované v DP.



Obr.7.21. Príklad transformácie adries pamäte na adresy fyzickej pamäte.

Virtuálnou pamäťou sa získa fiktívny pamäťový priestor s kapacitou V blokov, každý s vopred definovaným počtom k slov, resp. bajtov. Adresy týchto blokov sú 0, 2^k, 2^{k+1}, ..., 2^V. Pre používateľa to znamená, že každý blok s uvedenou adresou je dostupný procesoru, tzn. javí sa akoby bol fyzicky umiestnený v HP alebo v DP.

7.5.2 Spôsoby mapovania virtuálnej pamäte

Virtuálne adresy (*VA*) priestoru VP sú generované v priebehu *kompilácie*. Ich transformácia na *fyzické adresy* (*FA*) priestoru FP sa uskutočňuje v priebehu *spracovania programu*. Uvedená transformácia adries sa nazýva *mapovanie pamäte*, ktoré je definované funkciou:

$$f_t : \{VA\} \rightarrow \{FA\} \cup \{\varnothing\}$$

Mapovanie adries je časovou funkciou, pretože fyzická pamäť je dynamicky alokovaná a dealokovaná, čo formálne je vyjadrené vzťahom:

$$f_t(v) \langle \begin{array}{l} m \Leftarrow VP[v] \in FP[m] \\ 0 \Leftarrow VP[v] \notin FP[m] \end{array}$$

kde

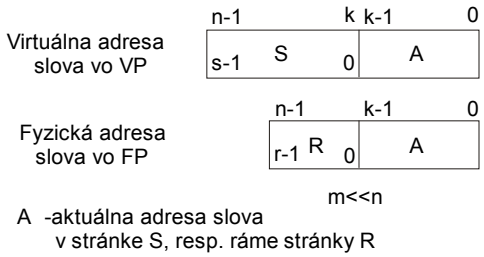
$m \in \{FA\},\ v \in \{VA\}$

Virtuálna adresa je adresa, ktorou sa identifikuje slovo (blok) programu v logickom adresovom priestore. Fyzická adresa predstavuje adresu kópie tohto slova (bloku) lokalizovaného v HP, resp. v DP. Existujú dve skupiny VP: s *pevnou dĺžkou* blokov, ktoré sa nazývajú *stránky* a s *premenlivou dĺžkou* blokov, ktoré sa nazývajú *segmenty*.

Stránková organizácia pamäte. Pri stránkovej organizácii pamäte adresy 0, 1, ..., 2^k - 1 sa vzťahujú na stránku 0, adresy 2^k , 2^k + 1, ..., 2^{2k} – 1 na stránku 1, atď. Skupiny susedných 2^k pamäťových buniek sa zlučujú do pamäťového priestoru, ktorý sa nazýva:

- *stránka* v prípade VP,
- *rám stránky* v prípade FP.

Na Obr. 7.22 sú uvedené formáty *VA* a *FA*, kde *S* je adresa *stránky* v *logickom* (*virtuálnom*) *adresovom priestore* (*VA*), *A* – adresa slova v stránke *S* a *R* – adresa fyzického *rámu stránky* (*fyzickej adresy*) v *adresovom priestore fyzickej pamäte*, z čoho vyplýva, že stránka je skupina 2^k slov, zatiaľ čo rám stránky je skupina 2^k susedných pamäťových buniek v HP.



Obr.7.22. Formáty virtuálnych a fyzických adries v stránkovo organizovanej virtuálnej pamäti.

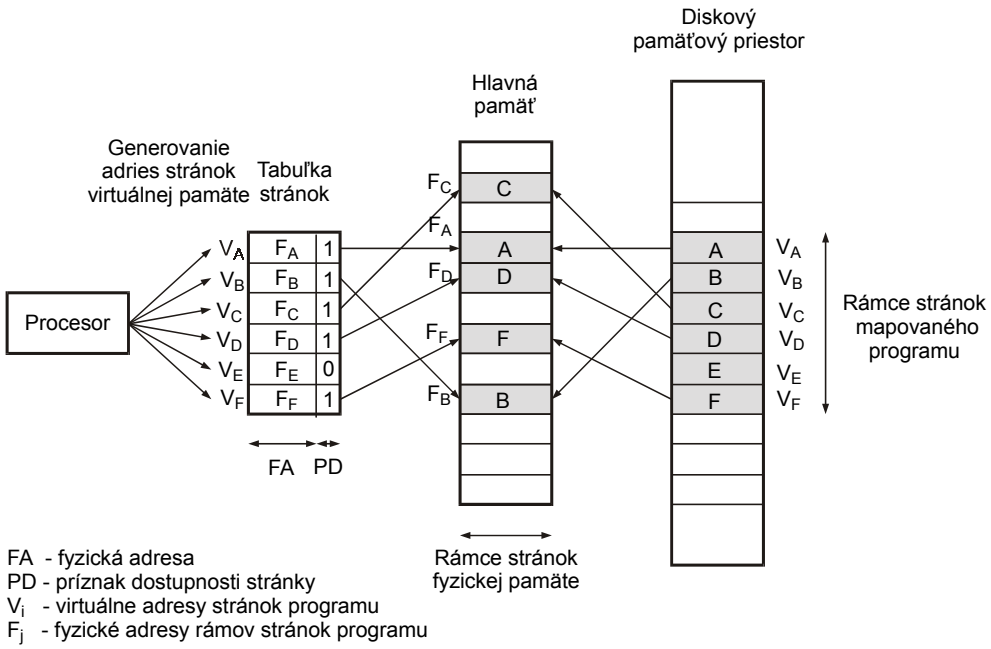
V ďalšom je prezentovaný model stránkovej organizácie pamäte, v ktorom FP je priestor HP a VP je priestor DP. V priebehu spracúvania informácií sa v rámci mapovacieho mechanizmu adresované stránky premiestňujú medzi HP a DP tak, aby bola v HP umiestnená tá stránka, ktorú v danom časovom intervale využíva procesor na vykonávanie programu. Priebežný stav pamäte ČP v procese mapovania je registrovaný pomocou *tabuľky stránok*. Jej základná funkcia vychádza z potreby priebežne registrovať prítomnosť kópií stránok premiestňovaných z priestoru VP do priestoru FP ako stránkových rámov a opačne. Preto je v nej zobrazovaný rad príznakov, ktoré tento proces

charakterizujú. V Tab. 7.3 je uvedený príklad tabuľky stránok, pozostávajúcej z M riadkov. Každej stránke v tabuľke stránok je priradený jeden riadok, v ktorom R_i označuje adresu i. rámu stránky (i = 1, 2, ..., M) lokalizovaného v HP, PD_i – príznak dostupnosti stránky S_i v HP (PD_i = 1: adresovaná stránka je prítomná v HP, v opačnom prípade sa v HP nenachádza), PA_i – príznak aktivity stránky (PA_i = 1: stránka bola aspoň raz využitá, PD_i = 0: stránka ešte nebola aktívna, príznak sa využíva pre rôzne stratégie vyraďovania nepotrebných stránok z HP) a PZ_i – príznak zmeny (PZ_i = 1: obsah stránky v HP bol zmenený, stránka je neplatná).

Tab.7.3.Informačný obsah tabuľky stránok.

R_0	PD_0	PA_0	PZ_0
R_1	PD_1	PA_1	PZ_1
…	…	…	…
R_i	PD_i	PA_i	PZ_i
…	…	…	…
R_M	PD_M	PA_M	PZ_M

Príklad systému virtuálnej pamäte pri jej stránkovej organizácii je uvedený na Obr. 7.23. Program mapovaný do HP pozostáva zo stránok A, B, C, D, E a F uložených v diskovom pamäťovom priestore. Ich virtuálne adresy VA sú V_A , V_B , V_C , V_D , V_E , V_F a fyzické adresy FA kópií týchto stránok v rámoch stránok hlavnej pamäte HP sú F_A , F_B , F_C , F_D , F_E , F_F . Pre zjednodušenie ilustrácie princípu stránkovej organizácie VP je z príznakov uvedený iba príznak dostupnosti stránky PD , ktorý identifikuje prítomnosť stránky v HP.



Obr.7.23. Princíp mapovania virtuálnej pamäte pri stránkovej organizácii jej pamäťového priestoru.

Na mapovanie virtuálnych adries na adresy fyzické je využitá tabuľka stránok, počet riadkov ktorej sa rovná počtu mapovaných adries blokov daného programu. Dĺžka VA generovaných procesorom je podstatne menšia ako dĺžka adries riadkov tabuľky stránok, v ktorých sú uložené

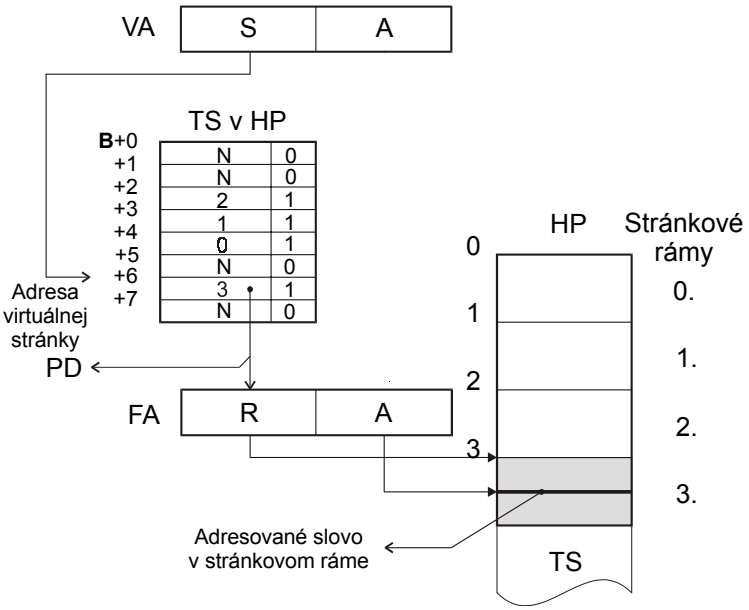
príslušné FA hlavnej pamäte. Napríklad, ak priestor VP je definovaný 32-bitovými adresami, t.j. predstavuje logický pamäťový priestor s kapacitou 4 *GB*, pozostávajúci z 256 *Kstránok*, každá s veľkosťou 16 *KB* ($2^{32-16} = 2^{16}$ = 64 *KB* = 256 *Kstránok*), potom kapacita tabuľky stránok je 1 *MB* za predpokladu, že každý jej riadok reprezentujú 4 *B* potrebné na zobrazenie fyzickej adresy, príznakov, prípadne ďalších doplňujúcich informácií (256 $k \times 4$ *B* = 1 *MB*). Za účelom redukcie kapacity tabuľky stránok sa využívajú rôzne metódy, ktoré kapacitu tabuľky stránok znižujú na počet stránok využívaných daným programom. Transformácia VA na adresy riadkov tabuľky stránok sa vo všeobecnosti uskutočňuje rôznymi hardvérovými a softvérovými metódami, z ktorých medzi základné patrí metóda hašovania adries. Hašovanie (transformácia kľúča) je metóda vyhľadávania, pri ktorej sa adresy hľadaných rámov stránok vo FP určujú z príslušných kľúčov – adries rámov VP. Formálne je definovaná funkcia

$h: K \rightarrow A$

z množiny kľúčov do množiny adries A. Pretože pre dĺžku K_b kľúča K a dĺžku A_b adresy A v bitoch platí $K_b > A_b$, vytvára sa tým predpoklad na výrazné zníženie kapacity tabuľky stránok. Riešenie uvedeného problému sa uskutočňuje v rámci mapovania VP.

Medzi základné spôsoby mapovania VP patrí:

- mapovanie prostredníctvom *tabuľky stránok TS* (*PT* - *P*age *T*ables),
- mapovnie prostredníctvom *pamäti preložených stránok PPS* (*TLB* - *T*ranslation *L*ookaside *B*uffer),
- mapovanie prostredníctvom *registrovej pamäte RP*.



Obr.7.24. Organizácia mapovania pomocou tabuľky stránok TS

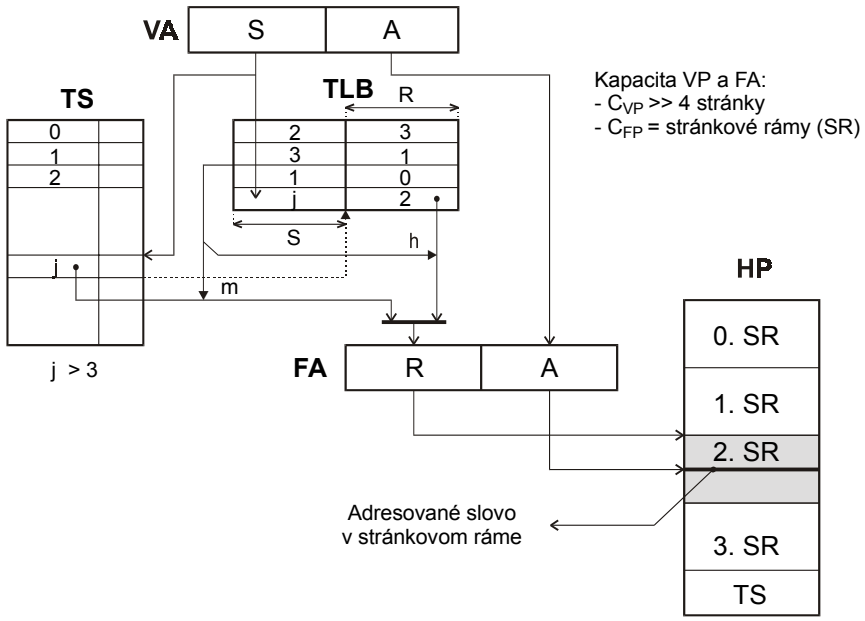
Na Obr. 7.24 je uvedený spôsob mapovania pamäte prostredníctvom TS. Ak je *B* pevne definovaná bázová adresa *TS* uloženej v HP, tak výsledná adresa (*R*) rámu stránky, zodpovedajúceho stránke s virtuálnou adresou (*S*) je:

$R = TS\,[(B) + S]$

pričom adresa stránky sa metódami hašovania transformuje do takej dĺžky, aby počet riadkov TS bol minimalizovaný. Nedostatkom mapovania pamäte VP pomocou TS je pomalé vykonávanie mapovania v dôsledku dvojnásobného prístupu do HP pri každom výpočte FA. Východisko poskytuje mapovanie prostredníctvom pamäte preložených adries (TLB) , v ktorej sú uložené adresy najčastejšie používaných stránok v práve prebiehajúcom intervale spracovania programu. Položka v TLB (asociatívna pamäť) má formát:

$$\langle TLB \rangle = \langle S \rangle \ \langle R \rangle$$

tzn. sú v nej uložené virtuálne adresy aktívnych stránok, tzn. tých stránok, ktoré boli v ostatnom čase najviac sprístupňované (kontrola sa uskutočňuje prostredníctvom špecializovaných príznakov nastavovaných pri aktivácii stránky). TLB sa asociatívne prehľadáva podľa kľúča, ktorým je virtuálna adresa VP.



Obr.7.25. Mapovanie vyrovnávacej pamäte VP pomocou tabuľky preložených adries TLB

Pri asociatívnom prehľadávaní TLB sú definované nasledujúce funkcie: *prítomnosť stránky* v TLB (*h* - *h*it) a výpadok stránky v TLB (*m* - *m*iss), kde

$$h \mid S \in TLB, \ \ m \mid S \notin TLB,$$

V prípade, ak adresovaná stránka sa nachádza vo FP, prečíta sa jej adresa z poľa R asociatívnej TLB a vytvorí sa formát *FA* v tvare

$$\langle FA \rangle :: = \langle R \rangle \langle A \rangle$$

Na Obr. 7.25 je uvedené mapovanie *j*. virtuálnej adresy, ktorá je prítomná v TLB, čo potvrdzuje svojou hodnotou $h = 1$, indikujúcej prítomnosť stránky v TLB funkciou $h \mid j \in TLB$. V tomto prípade $FA = 2 \# A$, kde 2 je druhý rám stránky v HP, reprezentujúci výsledok mapovania *j*. stránky do FP a *A*, offset adresovaného slova v 2. stránke. V opačnom prípade, keď svojou hodnotou $m = 1$ je indikovaná neprítomnosť stránky v TLB, uskutoční sa transformácia VA na FA prostredníctvom

TS, pričom príslušná VA a FA sa následne uložia v TLB, z ktorej môže byť FA rámu stránky priamo využitá pri nasledujúcom adresovaní VP.

Segmentová organizácia pamäte. Segmentová organizácia pamäte vyplýva z požiadavky vytvoriť v pamäťovom podsystéme kontinuálny priestor premenlivej dĺžky, ktorý môže byť uložený od ľubovolnej adresy virtuálneho priestoru. Spravidla sa používa v kombinácii so stránkovou organizáciou pamäte, čím sa získava *segmentovo-stránková* (viacúrovňová) organizácia virtuálnej pamäte. Jednotlivé segmenty pozostávajú z rozličného počtu stránok. Osobitný prípad segmentovej organizácie virtuálnej pamäte, v ktorej segmenty pozostávajú iba z jednej stránky, reprezentujú stránkovú organizáciu tejto pamäte.

Formát *VA* pri segmentovo-stránkovej organizácii má tvar (Obr.7.26):

$\langle VA \rangle :: = \langle SG \rangle \langle ST \rangle \langle PS \rangle$

kde

SG je adresa segmentu v logickom adresovom priestore VP,
ST - číslo stránky v segmente s nerovnakým počtom stránok,
PS - posuv slova v stránkach s rovnakým počtom slov.

Adresa *i*. segmentu generovaná procesorom sprístupňuje *i*. riadok v *tabuľke segmentov* (*TSG*), v ktorom je uložená vstupná adresa $ATST_i$ tabuľky stránok (TST), transformujúcej adresy stránok *i*. segmentu na rámy stránok fyzickej pamäte (hlavnej pamäte) pri vykonávaní príslušného programu. TSG je alokovaná v HP od bázovej adresy ATSG tabuľky segmentov. Adresovaný *j*. riadok stránky TST_i v *i*. segmente sa určí ako súčet

$$TST_i = ATST_i + ST_j$$

kde ST_j je číslo *j*. stránky v *i*. segmente. Prostredníctvom tohto riadku sa v súlade s funkciou TST transformuje adresa segmentovo-stránkovej organizácie priestoru VP do priestoru FP. Fyzická adresa *FA* je definovaná formátom

$\langle FA \rangle :: = \langle R_j \rangle \langle PS \rangle$

kde R_j je rám stránky v priestore FP. V prípade, keď rámy stránok pozostávajú z nerovnakého počtu slov, fyzická adresa slova je definovaná v tvare:

$\langle FA \rangle :: = \langle R_j \rangle + \langle PS \rangle$

t.j. schému generovania *FA* treba doplniť na výstupe ďalšou sčítačkou.

Ochrana virtuálneho priestoru prideleného *i*. programu sa uskutočňuje na základe vyhodnotenia dvoch ohraničení adresového priestoru definovaného vzťahom

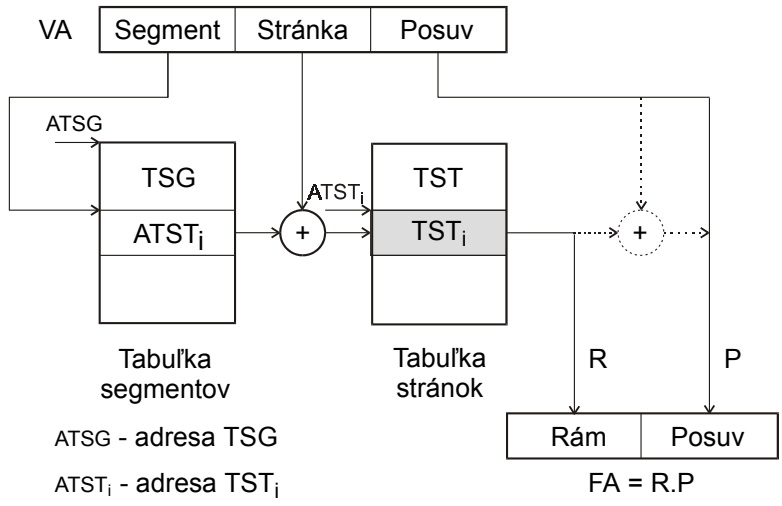
$$BA_i \leq A_i \leq HA_i$$

kde

BA_i je bázová adresa prideleného priestoru,
A_i - adresa v pamäťovom priestore pridelenom programu,
HA_i - hraničná adresa prideleného priestoru.

Uvedený kombinovaný prístup mapovania VP je pomerne pomalý, pretože vzhľadom na umiestnenie TSG a TST v HP vyžaduje 3 prístupy do HP pri každom pamäťovom odkaze. Tento

nedostatok sa obyčajne odstraňuje použitím asociatívnej pamäte na uloženie v TSG a TST iba najaktuálnejších údajov.



Obr.7.26. Mapovanie adresového priestoru v segmentovo-stránkovej organizovanej virtuálnej pamäte do fyzickej pamäte (hlavnej pamäte).

Pri vykonávaní programu, ktorému je vyhradený logický adresový priestor virtuálnej pamäte, sa realizujú dva základné mechanizmy riadenia tohoto procesu:

- mapovanie pamäte (prevod virtuálnych adries na fyzické adresy),
- premiestňovanie stránok medzi FP a VP.

Podobné mechanizmy sa uplatňujú na ľubovolnej úrovni pamäťového podsystému. Napríklad, na úrovni HP a pamäte cache sa uskutočňuje mapovanie blokov HP do rámov blokov pamäte cache podobným mechanizmom ako pri mapovaní stránok VP do rámov stránok HP. Analogicky možno hovoriť i o premiestňovaní stránok, resp. blokov príslušných úrovní pamäťového podsystému.

7.5.3 Metódy vyraďovania stránok

Ak vo FP (v HP) nie je umiestnená stránka z adresovaného priestoru VP, t.j. dochádza k výpadku stránky, čo je indikované nasledujúcim nastavením príznakov:

$m = 1 \Leftarrow S \notin$ TLB (mapovací mechanizmus využíva TLB)
alebo
$PD = 0 \Leftarrow S \notin TST[S\,] \,.\, PD$ (mapovací mechanizmus využíva TST),

je potrebné najsamprv vyradiť nepotrebný stránkový rám z HP a do uvolnenej pozície uložiť *kópiu adresovanej stránky*, t.j. vykonať postupnosť nasledujúcich operácií:

$VP\,[TLB\,[\gamma]\,.\,S\,] := HP\,[TLB\,[\gamma]\,.\,R]$ → vyradenie stránky z HP

$HP\,[TLB\,[\gamma]\,.\,R] := VP\,[VA]$ → zápis kópie stránky do HP

$TLB\,[\gamma]\,.\,S := VA$ → úprava TLB

kde γ je pozícia zápisu v TLB, v ktorej je uložená informácia o stránkovom ráme určenom na vyradenie z HP.

Medzi základné stratégie vyraďovania nepotrebných stránok z HP patria [161]:

- stratégia náhodného výberu,
- stratégia výberu podľa príznaku aktivity,
- stratégia LRU (Least Recently Used),
- stratégia FIFO (First In First Out).

Stratégia náhodného výberu. Realizuje sa pomocou generátora náhodných čísiel, ktorý určí riadok na vyradenie z HP.

Stratégia výberu podľa príznaku aktivity. Vyraďuje sa stránka, ktorá počas daného časového intervalu nebola ani raz aktivovaná. Obvodové riešenie vyraďovania stránok je založené na priradení každému rámu stránky, resp. rámu bloku v pamäti cache, *preklápacieho obvodu príznaku aktivity* nastavovaného signálom P_α, kde α je pozícia VA v tabuľke stránok (viď algoritmus riadenia cache a Obr. 7.9).

Stratégia LRU. Vyraďuje sa najdlhšie nepoužívaná stránka, na čo sa využíva počítadlo aktivácií PC_i nastavované príznakom aktivity P_i, kde $i = 0,1, \ldots, m$ je číslo stránkového rámu. Na základe

$$PC_i := \begin{cases} 0 \Leftarrow P_i = 1 \\ PC_i + 1 \Leftarrow P_i = 0 \end{cases}$$

sa vyradí najmenej aktívna stránka, ktorej PC_i je nastavené na najvyššiu hodnotu.

Stratégia FIFO. Vyraďuje sa stránka (blok), ktorá bola najdlhšie umiestnená v HP (resp. v cache). Funkcia PC je v tomto prípade definovaná vzťahom:

$$PC_i := \begin{cases} PC_i \Leftarrow P_i = 0 \\ PC_i + 1 \Leftarrow P_i = 1 \end{cases}$$

kde P_i je príznak zápisu stránky do HP (resp. bloku do cache).

Príklad 7.2. *Nech FP pozostáva z 5-och stránkových rámov: 0, 1, 2, 3, 4 a VP z 10-ich stránok: 0, 1, 2, ... , 9 a nech každá stránka reprezentuje skupinu 4-och slov, ktorých adresy vo virtuálnom priestore sú 4i, 4i +1, ..., 4i + 3 pre i = 0,1,2, ... , 9. Stav počítadiel stránok vyplývajúcich zo sekvencie adries slov sprístupňovaných vo VP pri spracúvaní daného programu je pre stratégie vyraďovania LRU A FIFO uvedený v tabuľke:*

Tab. 7.2. Príklad určenia stránky na vyradenie podľa stratégie LRU a FIFO

Adresa slova 4i+3	*Postupnosť stránok i*	*Stav počítadiel pri stratégii*									
		LRU					*FIFO*				
		PC_0	PC_1	PC_2	PC_3	PC_4	PC_0	PC_1	PC_2	PC_3	PC_4
0, 1, 2, 3	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
9	*2*	*1*	*2*	*0*	*2*	*2*	*2*	*0*	*1*	*0*	*0*
16, 17	*4*	*2*	*3*	*1*	*3*	*0*	*3*	*0*	*2*	*0*	*1*
5, 6, 7	*1*	*3*	*0*	*2*	*4*	*1*	*4*	*1*	*3*	*0*	*2*

8, 10, 11	*2*	*4*	*1*	*0*	*5*	*2*	*5*	*2*	*4*	*0*	*3*
18	*4*	*5*	*2*	*1*	*6*	*0*	*6*	*3*	*5*	*0*	*4*
7	*1*	*6*	*0*	*2*	*7*	*1*	*7*	*4*	*6*	*0*	*5*
8, 9, 11	*2*	*7*	*1*	*0*	*8*	*2*	*8*	*5*	*7*	*0*	*6*
12, 14	*3*	*8*	*2*	*1*	*0*	*3*	*9*	*6*	*8*	*1*	*7*
1, 2, 3	*0*	*0*	*3*	*2*	*1*	*4*	*10*	*7*	*9*	*2*	*8*
15	*3*	*1*	*4*	*3*	*0*	*5*	*11*	*8*	*10*	*3*	*9*
Vyradí sa rám		*4 ⇐ PC_4 = max*					*0 ⇐ PC_0 = max*				
Vyradenie nastáva po sprístupnení 24-ho slova z pamäte											

•

Príklad 7.3. *Zobrazte tabuľku premiestňovania stránok za predpokladu ,že kapacita FP je iba 3 stránkové rámy: 0,1,2 a poradie sprístupňovania stránok reprezentuje prúd:*

0, 1, 2, 5, 2, 3, 7, 2, 1, 3, 1

Porovnajte stratégie LRU, FIFO a OPT (optimálna stratégia, ktorá vyraďuje nepotrebnú stránku na základe znalosti všetkých aktivít stránok v budúcnosti) vyraďovania stránkových rámov a vypočítajte koeficient úspešnosti

$k_h = h / p$

kde h je počet úspešných prístupov k stránkam vo FP z celkového počtu prístupov p = h + m.

Tab. 7.3. Tabuľka premiestňovania stránok

Prúd aktivít stránok	*Stratégie premiestňovania stránok*									*Vyradenie stránok*
	LRU			*FIFO*			*OPT*			
	0	1	2	0	1	2	0	1	2	
0	0			0			0			
1	0*	1		0●	1		0	1		
2	0**	1*	2	0●●	1●	2	0	1	2	
5	5	1**	2*	5	1●●	2●	5	1	2	
2	5*	1***	2	5●	1●●●	2●●	5♦	1	2	*LRU,FIFO,OPT*
3	5**	3	2*	5●●	3	2●●●	3	1	2	
7	7	3*	2**	5●●●	3●	7	7	1	2	
2	7*	3**	2	2	3●●	7*	7♦	1	2	*LRU, OPT*
1	7	1	2*	2●	1	7**	7	1	2♦	*OPT*
3	3	1	2**	2●●	1●	3	3	1	2♦	
1	3	1	2***	2●●●	1●●	3	3♦	1	2♦	*LRU,FIFO,OPT*
k_h	3/11			2/11			4/11			

Označenie stránok určených na vyradenie:

- * *počet časových jednotiek, v priebehu ktorých je stránka prítomná v HP od poslednej aktivácie (vyraďuje sa stránka s najväčším počtom časových jednotiek),*
- • *počet časových jednotiek, v priebehu ktorých je stránka prítomná v HP,*
- ♦ *počet časových jednotiek, po uplynutí ktorých bude v budúcnosti sprístupnenie danej stránky požadované (optimálna stratégia, ktorá má iba teoretický význam).*

Z analýzy vyraďovania stránok z HP vyplýva, že v uvedenom príklade pre stratégie FIFO, LRU a OPT (teoretická stratégia vyraďovania na základe predikcie ich využívania) koeficienty úspešnosti sú:

$$k_{FIFO} < k_{LRU} < k_{OPT}$$

•

7.5.4 Konzistencia údajov v pamäti

Počas vytvárania kópií operandov vo VP môže nastať situácia, keď obsah originálu v HP nebude v zhode s obsahom kópie vo VP. K tomu dochádza najmä pri zápise nových údajov z diskovej pamäte (DP) do HP a pri zápise nových údajov z procesora do VP.

Na zabezpečenie zhody údajov sa kópie jednotlivých údajov vo VP označujú príznakmi platnosti (PP_i) a zmeny (PZ_i) údajov, ktoré sa nastavujú takto:

$$PP_i := \langle \begin{matrix} 1 \Leftarrow VP[A_i] = HP[A_i] \\ 0 \Leftarrow VP[A_i] \neq HP[A_i] \end{matrix}$$

$$PZ_i := \langle \begin{matrix} 1 \Leftarrow ZS/\overline{CT}.\, ZS_i = 1 \\ 0 \Leftarrow CT.ZS_i = 1 \end{matrix}$$

kde

A_i je adresa kópie údaja v *i.* pozícii *VP*,

ZS_i - riadiaci signál zápisu údaja do *i*. pozície *VP* ($i = \alpha \vee \gamma$),

$ZS/\overline{CT}$ - signál riadenia zápis a čítania v v úrovni HP-VP pamäťového podsystému.

Zhoda údajov medzi dvoma pamäťovými úrovňami M_1 a M_2 sa zabezpečuje prostredníctvom nastavenia hodnoty príznaku platnosti údajov PP_i:

$$PP_i := \langle \begin{matrix} 1 \Leftarrow M_1[S_1] = M_2[S_2] \\ 0 \Leftarrow M_1[S_1] \neq M_2[S_2] \end{matrix}$$

Medzi základné metódy na dosiahnutie zhody údajov patria:

- súčasný zápis údajov do M_1 a M_2, čím sa umožňuje zhoda údajov v ľubovolnom okamihu za cenu určitého spomalenia činnosti aktivácie pamäťového systému,
- pri zápise údaju do M_1 na miesto určené k vyradeniu sa príslušná stránka vždy vyradí do M_2 bez ohľadu nato či je to potrebné alebo nie je (ide o obnovu obsahu po každej výmene, ktorá spomaľuje činnosť pamäťového podsystému, poskytuje však jednoduchý riadiaci mechanizmus označovaný tiež ako „write through“),
- pri zápise údaju do M_1 na miesto určené k vyradeniu sa príslušná stránka vyradí do M_2 iba vtedy, keď PP_i indikuje nezhodu medzi originálom v M_2 a jeho kópiou v M_1, čo sa uskutočňuje na základe nastavenia príznaku zmeny (PZ_i) na jedničkovú hodnotu po predchádzajúcom zápise údajov z procesora do VP (ide o obnovu obsahu pamäte iba pri
- vyraďovaní nepotrebných údajov z M_1 do M_2, riadiaci mechanizmus ktorej je označovaný ako „write-back“). Pri vytváraní kópie do VP je tento príznak vždy nulový (je súčasťou ucelenej informácie o kópii v adresári VP).

Pri zápise údajov z diskovej pamäte (DP) do HP, keď môže vznikať nezhoda údajov sa kontroluje, či z niektorých zapisovaných blokov nie sú vytvorené kópie v DP. Keď tieto kópie

existujú, zruší sa ich obsah prostredníctvom nulovania príznaku platnosti ($PP_i = 0$) týchto blokov v DP. Následne musí procesor čítaním z HP zabezpečiť korektnosť obsahov kópií v DP.

7.6 Výkonnosť pamäťového podsystému

Výkonnosť hierarchicky usporiadaného pamäťového podsystému vyjadruje efektívny prístupový čas T_e do jeho ľubovoľnej úrovne. T_e závisí od hodnoty:

- koeficientu úspešnosti a
- frekvencií prístupu

k jednotlivým úrovniam pamäťového podsystému. Ak M_i a M_{i-1} sú dve úrovne hierarchicky usporiadaného pamäťového podsystému pre $i = 1,2, ..., n$, potom *koeficient úspešnosti* h_i (*h*it) má význam pravdepodobnosti, že sprístupňovaná informácia sa nachádza v úrovni M_i pamäťového podsystému. Skutočnosť, že sprístupňovaná informácia sa v úrovni M_i nenachádza je vyjadrená *koeficientom neúspešnosti* $m_i = 1 - h_i$ (*m* - miss). Napríklad, keď v dvojúrovňovom pamäťovom podsystéme (VP, HP) na každých N prístupov do VP pripadá jeden prístup do HP, potom efektívny prístupový čas k pamäťovému podsystému je

$$T_e = \frac{(N-1)T_V + T_H}{N} \tag{7.2}$$

kde T_V a T_H sú príslušné prístupové časy VP a HP. Zavedením koeficientu neúspešnosti:

$m = 1/N$

resp. úspešnosti

$$h = 1 - m = (N - 1) / N \tag{7.3}$$

je možné vzťah (7.2) upraviť do tvaru:

$$T_e = \quad hT_V + (1 - h)\, T_H \tag{7.4}$$

Pretože prenos údajov medzi VP a HP sa uskutočňuje po blokoch, v prípade aktivácie HP je nutné vo vzťahu (7.4) vyjadriť koeficientom b *podiel slov zmenených vo VP*, ktoré je nutné v rámci tejto aktivácie premiestniť do HP. Výsledný tvar vzťahu na výpočet efektívneho prístupového času pamäťového podsystému vytvoreného VP a HP potom je:

$$T_e = \quad hT_V + (1 - h)\,)\,(1 + b)\, T_H \tag{7.5}$$

Príklad 7.4. *Vypočítajte efektívny prístupový čas dvojúrovňového pamäťového podsystému, pozostávajúceho z vyrovnávacej pamäte a hlavnej pamäte, ktorých prístupové doby sú:*

$T_V = 0.1\ \mu s,\ \ T_H = 1\ \mu s,$

a koeficienty úspešnosti v prvej úrovni pamäťového podsystému a podielu slov zmenených vo vyrovnávacej pamäti:

$h = 0.95,\ b = 0.25$

Riešenie: Substitúciou hodnôt uvedených parametrov do vzťahu (7.5) dostávame:

$T_e = 0.95 \cdot 0.1 + (1 - 0.95)\ (1 + 0.25) \cdot 1 \cong 0.16\ \mu$

z čoho vyplýva, že efektívny prístupový čas hierarchicky usporiadaného pamäťového podsystému sa oproti prístupovému času HP skracuje približne 6-krát.

•

Uvedený prístup je možné zovšeobecniť pre *n*-úrovňový pamäťový podsystém na základe rozloženia pravdepodobnosti úspešného prístupu k jednotlivým úrovniam, pre ktoré platí:

$$h_1 + (1 - h_1)\,((h_2 + (1 - h_2)\,(h_3 + \ldots + (1 - h_{n-1}) \ldots)) = 1$$

Vzhľadom na definíciu koeficienta úspešnosti (7.3) je možné vyjadriť frekvencie f_i prístupu k jednotlivým úrovniam pamäťového podsystému následovne:

$$\begin{aligned}
&f_1 = h_1 \\
&f_2 = (1 - h_1)\,h_2 \\
&f_3 = (1 - h_1)\,(1 - h_2)\,h_3 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&f_i = (1 - h_1)\,(1 - h_2) \ldots (1 - h_{i-1})\,h_i \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&f_n = (1 - h_1)\,(1 - h_2) \ldots (1 - h_{n-1}) \cdot 1
\end{aligned} \qquad (7.6)$$

čo vyplýva zo skutočnosti, že:

$$\sum_{i=1}^{n} f_i = 1$$

Situácia, pri ktorej sa sprístupňované slovo (blok, stránka) v *i*-tej úrovni nenachádza, sa nazýva *výpadok slova* (bloku, stránky). Prístupový čas t_i sprístupňovania slova v *i*-tej úrovni, po jeho výpadku v úrovni predchádzajúcej a vyjadrenie frekvencií prístupu k postupnosti jednotlivých úrovní pamäťového podsystému v tvare (7.6) umožňujú definovať vo všeobecnosti efektívnu prístupovú dobu hierarchického pamäťového podsystému v tvare:

$$T_e = \sum_{i=1}^{n} f_i \,.\, t_i = h_1 t_1 + (1 - h_1)\,h_2 t_2 + (1 - h_1)\,(1 - h_2)\,h_3 t_3 + \ldots + (1 - h_1) + (1 - h_2) \ldots (1 - h_{n-1})\,t$$

pričom platí:

$$f_1 >> f_2 >> \ldots >> f_n$$

$$t_1 < t_2 < \ldots < t_n$$

Celková cena (*C*) pamäťového podsystému je

$$C = \sum_{i=1}^{n} C_i \cdot c_i$$

kde C_i a c_i sú cena a kapacita *i*-tej úrovne pamäťového podsystému, pre ktoré platí:

$$C_1 > C_2 > \ldots > C_n \text{ a } c_1 < c_i < \ldots < c_n$$

Príklad 7.5. *V trojúrovňovom pamäťovom podsystéme (Cache, HP, DP) určite potrebnú prístupovú dobu T_H do HP za predpokladu, že efektívny prístupový čas do pamäťového systému má byť $T_e = 10$ μs a parametre jednotlivých pamätí sú nasledovné:*

Cache - $T_C = 25$ ns, $C_C = 512$ KB, $h_C = 0,98$, $C_C = 1,25$ cena / kB
HP- $c_H = 32$ MB, $h_H = 0,9$, $T_H = ?$ $C_H = 0,2$ cena / kB
DP- $T_D = 4$ ms ($h_D = 1$), $c_D = ?$ $C_D = 0,0002$ cena / kB

Efektívny prístupový čas do pamäťového systému vypočítame podľa vzťahu:

$$T_e = h_C T_C + (1 - h_C) h_H T_H + (1 - h_C)(1 - h_H) h_D T_D \leq 10$$

Po dosadení dostaneme:

$$10x10^{-6} = 0,98x25x10^{-9} + 0,02x0,9xT_H + 0,02x0,1x1x4x10^{-3}$$

odkiaľ

$T_H = 903$ ns

$C = C_C \cdot c_C + C_H \cdot c_H + C_D \cdot c_D \leq 15\,000$

$C_D = 39,8$ GB

•