# Kapitola 5

# Paměti a jejich organizace

## 5.1 Vnitřní a vnější paměti, vlastnosti jednotlivých typů

**Vnější paměti**

Jsou umístěny mimo základní jednotku. Lze je zařadit mezi periferní zařízení. Zápis a čtení se provádí stejným způsobem jako zápis a čtení na ostatní periferní zařízení.

Rozeznáváme magnetické páskové, diskové, optické atp.

**Vnitřní paměti**

Jsou částí základní jednotky. Patří mezi ně hlavní paměť a registry, případně další paměti uvnitř procesoru.

**Paměti můžeme dále dělit:**

- podle fyzikálního principu:
  - polovodičové
  - magnetické
  - optické
- podle způsobu výběru datových položek:
  - s adresovým výběrem (adresové)
  - s postupným výběrem (sériové)
  - asociativní - výběr podle klíče
  - zásobník LIFO
  - fronta FIFO
- podle možností a způsobů změny uložení informace:
  - paměti pro čtení i zápis RWM
    - obsah lze libovolně přepisovat RAM (je energeticky závislá. Po vypnutí napájení uložená informace zanikne)
- paměti permanentní
  - obsah určen při výrobě ROM

  - obsah je jednorázově naprogramován PROM

- paměti semipermanentní
  - obsah je určen naprogramováním, lze ho ale vymazat a přeprogramovat EPROM

**Typy pamětí**

- registry - klopné obvody
- vyrovnávací paměť - statická RAM
- hlavní paměť - dynamická RAM
- vnější paměť - pevný disk
- záložní paměť - optický disk, magnetická páska

**Statická paměť RAM** paměťová buňka je realizována jako bistabilní klopný obvod - v CMOS logice jako dvojice invertorů. Na jeden klopný obvod je potřeba 6 tranzistorů.

**Dynamická paměť RAM** jednotranzistorová paměťová buňka, data jsou uchována ve formě náboje na paměťovém kondenzátoru.

- *zápis* - paměťový kondenzátor se nabije nebo vybije
- čtení - dochází k vybíjení kondenzátorů ⇒ čtení je destruktivní a přečtená informace se musí znovu zapsat.

Kondenzátory se po určité době samovolně vybíjejí ⇒ v pravidelných cyklech se proto musí provádět obnova, tj. čtení a následně zápis. Na jeden bit této paměti stačí jeden kondenzátor.

Dynamické paměti jsou při stejné kapacitě podstatně levnější než statické. Dynamická paměť musí obsahovat obvody pro pravidelné obnovování, proto jsou tyto paměti pomalejší než statické a mají větší spotřebu v klidovém stavu, protože při obnovování dochází neustále k nabíjení a vybíjení paměťových kondenzátorů.

## 5.2 Základní parametry pamětí

**Paměťová buňka** - základní blok paměti, slouží k záznamu jednoho bitu.

**Paměťové místo** - skupina paměťových buněk, které lze současně zapisovat nebo číst.

**Položka** - obsah paměťového místa.

**Adresa** - číselné označení paměťového místa. Počet položek = *kapacita paměti* (množství informací, které lze do paměti uložit. $1K = 2^{10}$, $1M = 2^{20}$, $1G = 2^{30}$, $1T = 2^{40}$).

**Vybavovací doba** - časový interval, který uplyne od vyslání požadavku na čtení z paměti do okamžiku, kdy jsou data přečtena a jsou k dispozici.

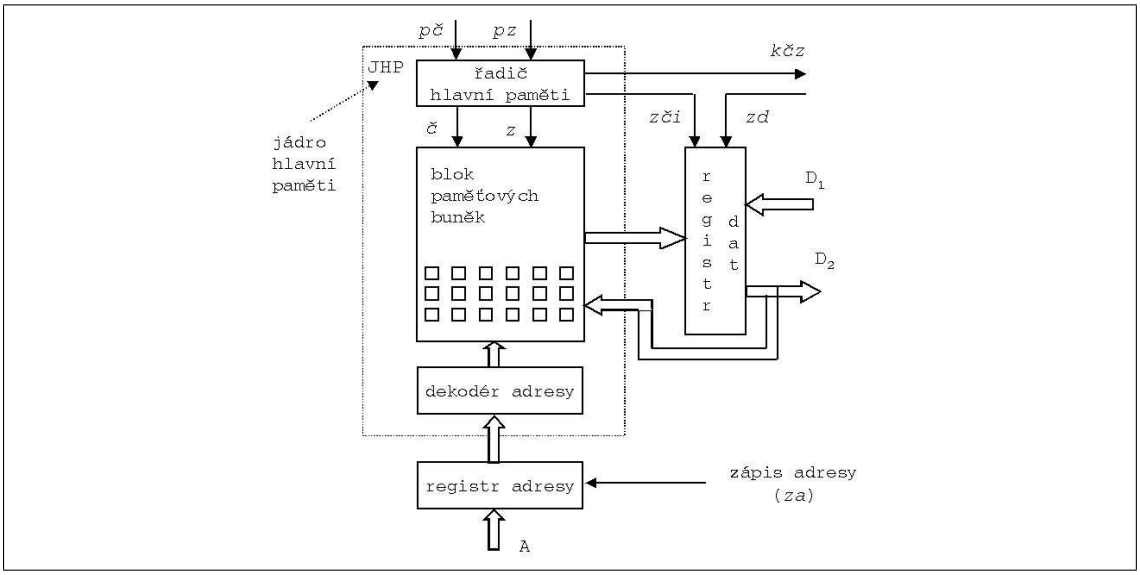
**Střední vybavovací doba** - průměrná vybavovací doba při typickém použití paměti.

**Doba zápisu** - obdoba vybavovací doby.

**Cyklus paměti** - minimální časový interval, který musí uplynout mezi dvěma po sobě jdoucími požadavky na činnost paměti (čtení nebo zápis)

## 5.3 Hlavní (operační) paměť

Hlavní paměť slouží pro uložení programu i dat (von N.). Paměťovým místům jsou přiřazena nezáporná celá čísla $0, \ldots, N-1$, kde $N$ je kapacita paměti. Číslům se říká adresa. Adresovým prostorem je množina všech možných adres a bývá určen počtem bitů v adrese.




Obrázek 5.1: schéma hlavní paměti

Na obrázku 5.1 je znázorněno schéma hlavní paměti. Před čtením nebo zápisem je potřeba vložit adresu do *registru adres*. Adresa se přivede na vstup $A$ a vyšle se řídící signál *za* (zápis adresy). Před zápisem je navíc potřeba uložit data do *registru dat*. Data se přivedou na vstup $D_1$ a vyšle se signál *zd* (zápis dat). Nyní je možné poslat požadavek na čtení nebo zápis pomocí *pč* (požadavek čtení) nebo *pz* (požadavek zápis). Jádro hlavní paměti provede požadovanou operaci a její ukončení ohlásí stavovým signálem *kčz* (konec čtení/zápis). Bylo-li požadováno čtení, jsou nyní data na výstupu $D_2$.

Řadič hlavní paměti řídí činnost paměti. Přijímá zvenčí požadavky na čtení a zápis (*pč*, *pz*), hlásí ukončení požadované činnosti *kčz* a řídí činnost bloku buněk. Dál zajišťuje signálem *zči* zápis čtené informace do *registru dat*.

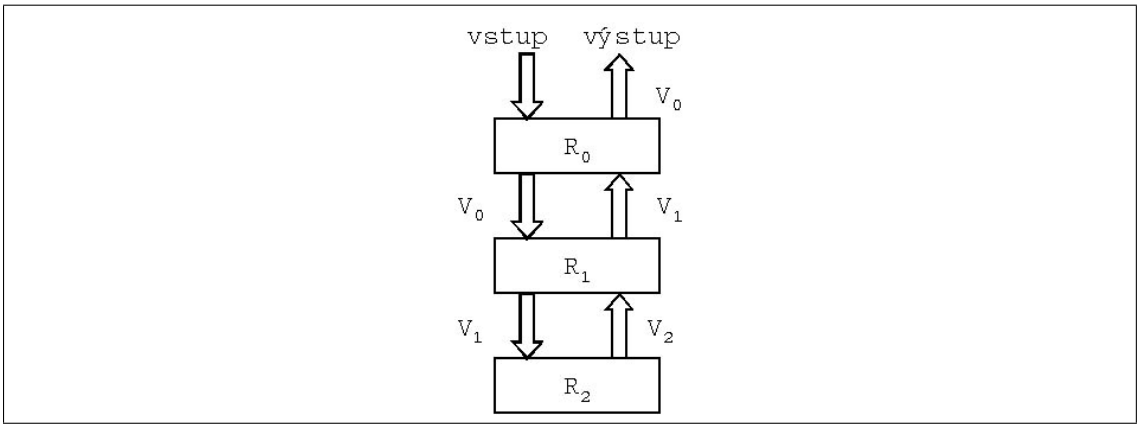
Výstup *registru adres* se v *dekodéru adres* převede do kódu požadovaného pro výběr paměťového místa v *bloku buněk* (např. kód 1 z N $\Rightarrow$ každému paměťovému místu přísluší právě jeden výstup obvodu *dekodér adres*).

## 5.4 Paměti typu LIFO a FIFO a jejich realizace

### 5.4.1 LIFO

Jedná se o zásobníkovou paměť. Paměťové místo, na které se informace ukládá, se nezadává - je určeno konstrukcí a stavem paměti. Nezadává se ani paměťové místo, ze kterého se má číst. Je to místo, které uchovává posledně uloženou položku.

Na obrázku 5.2 je znázorněné jedno z možných schémat zásobníkové paměti. $R_0 \ldots R_2$ - registry. Při zápisu se přepíše $R_1$ do $R_2$, $R_0$ do $R_1$ a zapisovaná informace do $R_0$. Při čtení se přečte informace z $R_0$ a vše se posune nahoru. $R_0$ je vrchol zásobníku.



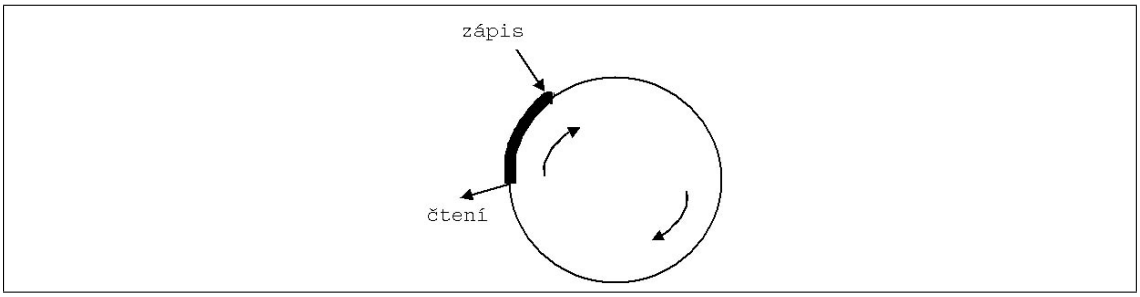

Obrázek 5.2: paměť LIFO

Paměť LIFO může být také simulována v hlavní paměti programově nebo technickými prostředky. Pokud je realizována technickými prostředky, používá se registr *ukazatel zásobníku* (*stack pointer*). Jeho obsah se před zápisem snižuje a po čtení se zvyšuje.

### 5.4.2 FIFO

Jedná se o paměť typu fronta. Vždy se z ní čte položka, která je v ní nejdéle. Využívá se jako vyrovnávací paměť.

U této paměti nestačí jedno ukazovátko na vrchol, ale jsou potřebné dvě. Na začátku jsou obě ukazovátka nastavena stejně. Při zápisu se posouvá jedno ukazovátko - určuje, kam se má zapsat další položka. Při čtení se posouvá druhé ukazovátko a určuje odkud se má číst další položka. Obě ukazovátka se posouvají stejným směrem, a to cyklicky - viz. obrázek 5.3. Ukazují-li obě na stejné místo, pak je paměť úplně prázdná nebo úplně plná. K odlišení se používá jednobitová paměť, která se nastavuje do jedničky při každém zápisu a která se nuluje, když dojde ke shodě ukazovátek v důsledku čtení.




Obrázek 5.3: paměť FIFO

## 5.5 Asociativní paměti a jejich realizace

Paměťové místo se u asociativní paměti při čtení vybírá podle zapsané informace. Položka, která je v paměti zapsána obsahuje tři části $\begin{cases} \text{data} \\ \text{klíč} \\ \text{bit platnosti} \end{cases}$

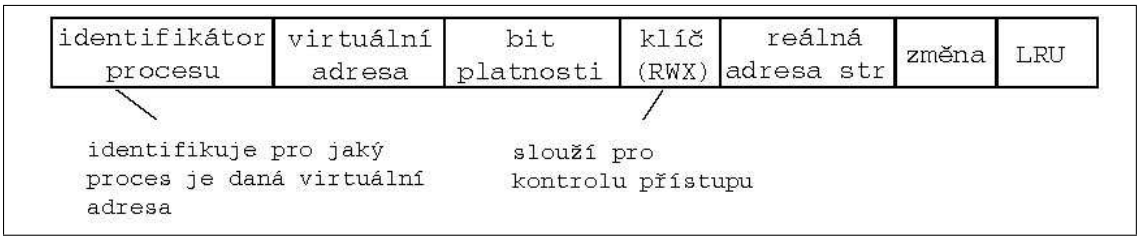
Paměťová místa s nulovým bitem platnosti se považují za neobsazená. Při čtení z této paměti se místo adresy zadává klíč. Je-li v paměti položka se zadaným klíčem a bitem platnosti nastaveným na 1, přečte se, jinak se hlásí, že položka v paměti není.

Která část položky je klíčem a která je daty, může být určeno pro danou paměť pevně nebo může být zadáno spolu s klíčem pomocí *masky klíče*.

Zápis se provádí do jednoho z neobsazených paměťových míst. Neexistuje-li neobsazené paměťové místo a má se provést zápis, vybere se podle vhodného algoritmu (LRU, NFU apod.) některé obsazené místo a přepíše se.

Asociativní paměť bývá tvořena registry a srovnávacími obvody mezi všemi klíči v paměti a zadávaným klíčem. Toto řešení je ovšem nákladné ⇒ paměti mají malou kapacitu, ale rychlou vybavovací dobu. Využívají se proto jako rychlá vyrovnávací paměť.

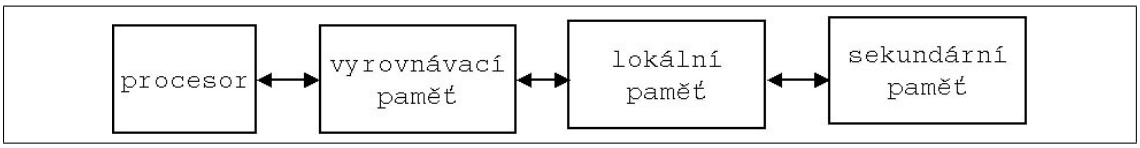
Plně asociativní paměť se používá pro zrychlení překladu virtuálních adres na reálné. Tato paměť se označuje jako **TLB** (**T**ranslation **L**ookside **B**uffer - vyrovnávací paměť pro nahlížení při překladu). Jedná se tedy pouze o pomůcku, která zrychluje adresaci. V adresáři asociativní paměti je každé ze stránek, která má být takto adresována, přiřazeno jedno slovo rozdělené do dvou částí. V jedné části je virtuální adresa stránky a v druhé je její reálná adresa. Každý bit adresáře je vybaven srovnávacím obvodem, který signalizuje shodu nebo neshodu s odpovídajícím bitem hledané adresy. Pokud je hledání v asociativní paměti neúspěšné, spustí se algoritmus překladu používající úplný adresář uložený v HP. Na obrázku 5.4 je znázorněna struktura slova v plně asociativní paměti.




Obrázek 5.4: struktura slova plně asociativní paměti

## 5.6 Vyrovnávací paměť (cache), její struktura a adresace

Hlavní paměť je většinou příliš pomalá. Proto se mezi procesor a HP vkládá další úroveň paměti - vyrovnávací paměť. Tím vzniká trojúrovňový paměťový systém - viz. obrázek 5.5.




Obrázek 5.5: trojúrovňový paměťový systém

Vyrovnávací paměť (VP) bývá až desetkrát rychlejší než hlavní paměť (HP) ⇒ zrychlení výpočtu. Programátor ale nemůže st touto pamětí pracovat. Přesuny dat mezi VP a HP řídí automaticky řadič počítače.

VP má pouze zlomek kapacity HP. Jsou zde uloženy duplikáty obsahu některých vybraných míst z HP. Adresy míst uložených v VP jsou zapsány v adresáři, podle jehož obsahu lze zjistit, zda hledaná informace je ve VP nebo není.

Pokud se nenajde hledaná adresa ve VP, přečte se informace z HP a přesune se do procesoru. Při tom se současně uloží do VP. Navíc se do VP ukládá ještě několik sousedních adres ⇒ přesouvání bloku.

U vyrovnávacích pamětí se nepoužívá plně asociativní paměť, ale paměť s omezeným stupněm asociativity.

#### Adresář VP s omezeným stupněm asociativity

Stupeň asociativity udává, kolik slov adresáře je nutné prohledat při hledání určitého bloku ve VP. Pro každý blok je tedy vyhrazena skupina *rámů*, do nichž smí být zapsán, tzv. *sada rámů*. Do každé sady ve VP lze zapsat jen určitou podmnožinu bloků z HP, tzv. *třída*. *Třídy* se volí tak, aby bloky, které za sebou následují v HP, patřily do různých tříd. Tj. každé položce z HP je podle její adresy přiděleno jedno pevné místo (případně několik míst), kde se ve VP může nacházet. Adresář cache lze pak realizovat běžnou pamětí RAM. Přítomnost položky se zjistí porovnáním požadovaného klíče s klíčem uloženým v adresáři.

#### Strategie výměny dat mezi VP a HP

Používají se stejné strategie jako pro uvolňování rámů stránek v HP, tj. LRU, FIFO a náhodný výběr. Pro VP se dává přednost takové strategii, kterou lze snadno realizovat obvodově.

**LRU** - používá se samostatný čítač u každého z rámů v sadě. Zápis i kontrola čítačů se provádí obvodově. Při každém volání bloku se jeho čítač vynuluje, ostatním se přičítá jednička (trestný bod za pasivitu). Vyřazuje se vždy ten blok, jehož čítač má největší hodnotu.

Při použití VP vzniká také problém v souvislosti s požadavkem zajištění shody dat zapsaných současně do VP a HP. Současné zapisování do VP a HP se nepoužívá, protože je dost pomalé (musí se čekat až se zapíše do HP). Častěji se používá řešení, kdy procesor zapisuje data jen do VP a změny se přenášejí do HP v okamžiku vyřazení příslušného bloku (write back). Tato strategie má tři modifikace:

- úklid bloku při každém vyřazení
- úklid bloku podle příznaku změny (pokud v bloku došlo ke změně)
- úklid bloku podle příznaku změny přes pomocnou paměť bloku

## 5.7 Virtuální paměť stránkovaná i segmentovaná; dynamický překlad adres

Virtuální paměť je systém několika pamětí s různými parametry, který je řízen tak, aby vytvářel adresové prostory potřebné velikosti pro programy a pro data.
* Virtuální paměť umožňuje:

- realizaci jednoho nebo několika virtuálních adresových prostoru, z nichž každý může být větší než je skutečná kapacita HP.
- úsporné využití HP tím, že jsou v ní přítomny pouze ty části programu a dat, se kterými procesor právě pracuje.

- vzájemnou ochranu jednotlivých programů v paměti a ochranu dat před neoprávněným přístupem a modifikací.

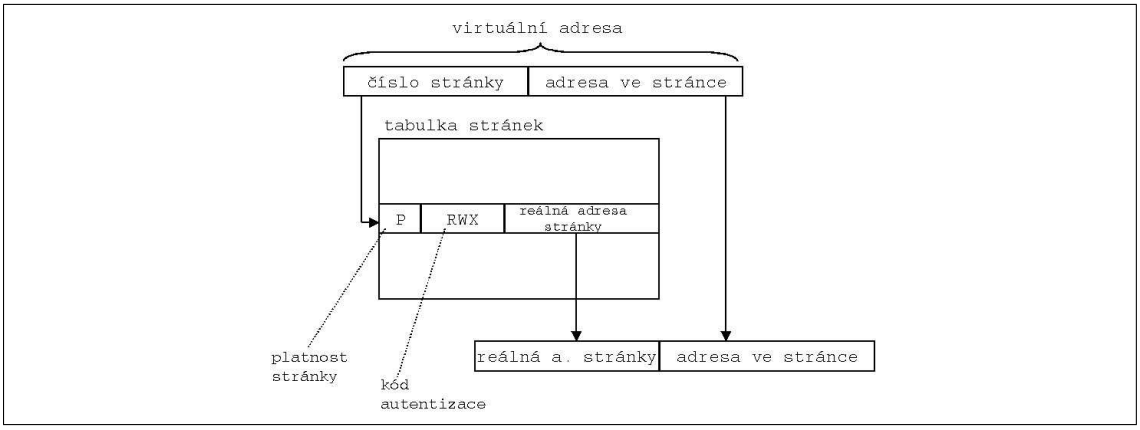
V režimu virtuální paměti pracuje program vždy s logickými (virtuálními) adresami. HP se adresuje fyzickými adresami ⇒ musí se překládat virtuální adresa do fyzické. VP tedy předstírá, že máme k dispozici podstatně větší paměťový prostor než je ve skutečnosti.

### 5.7.1 Stránkovaná paměť

Logický (virtuální) prostor je rozdělen na úseky pevné délky - *stránky*. Fyzický adresový prostor je rozdělen také na stejně velké úseky.

Logický prostor je realizován ve vnější paměti. Data se přesouvají do HP po jednotlivých stránkách, jsou-li v průběhu výpočtu požadována a pokud nejsou v HP.

Stránkovací mechanismus pracuje s datovou strukturou - *tabulka stránek* - uloženou v HP. Každé stránce odpovídá jedna položka v tabulce stránek - viz. obrázek 5.6




Obrázek 5.6: stránkovaná paměť

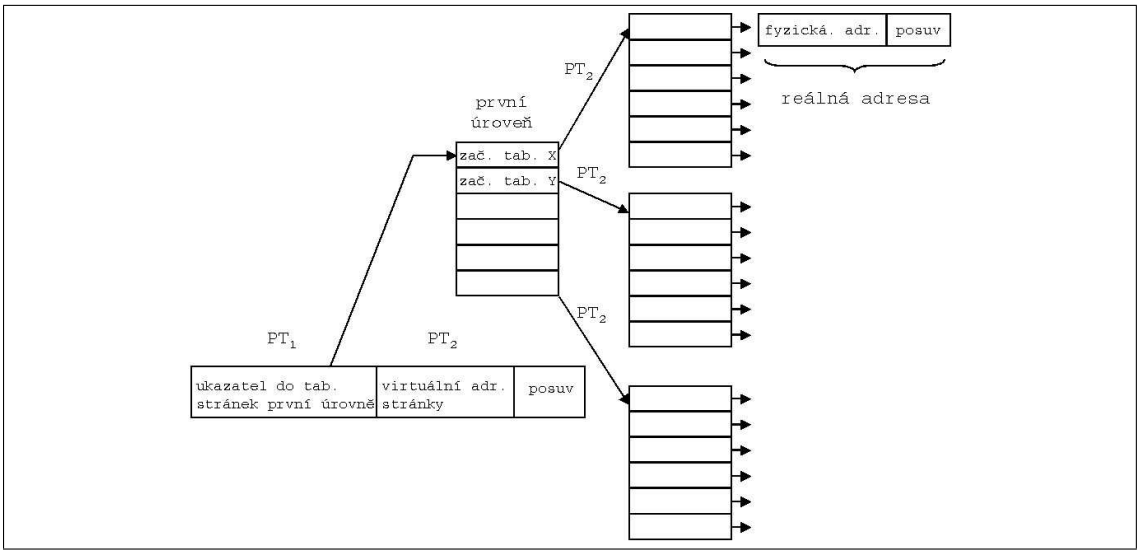
Problémem je, že tabulka stránek musí obsahovat jednu položku pro každou stránku i když tato stránky není použita ⇒ plýtvání místem. Řešením je dvouúrovňová organizace tabulky stránek - uchovávají se pouze používané stránky - viz. obrázek 5.7.

### 5.7.2 Stránkovaná segmentovaná paměť

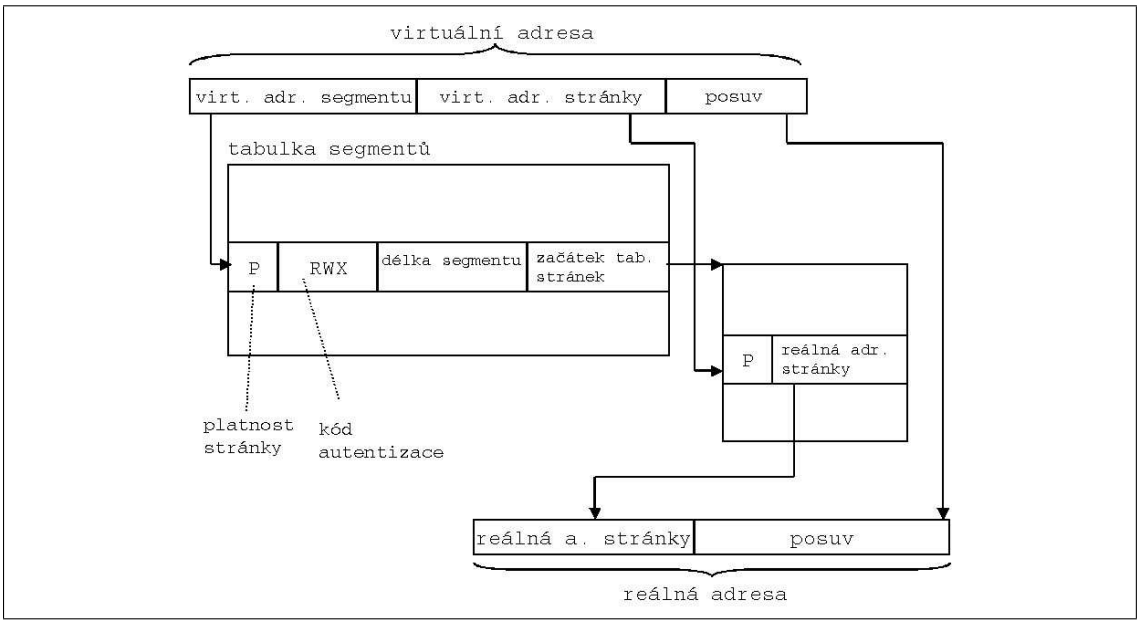
Na obrázku 5.8 je znázorněna tato paměť.

### 5.7.3 Segmentovaná paměť

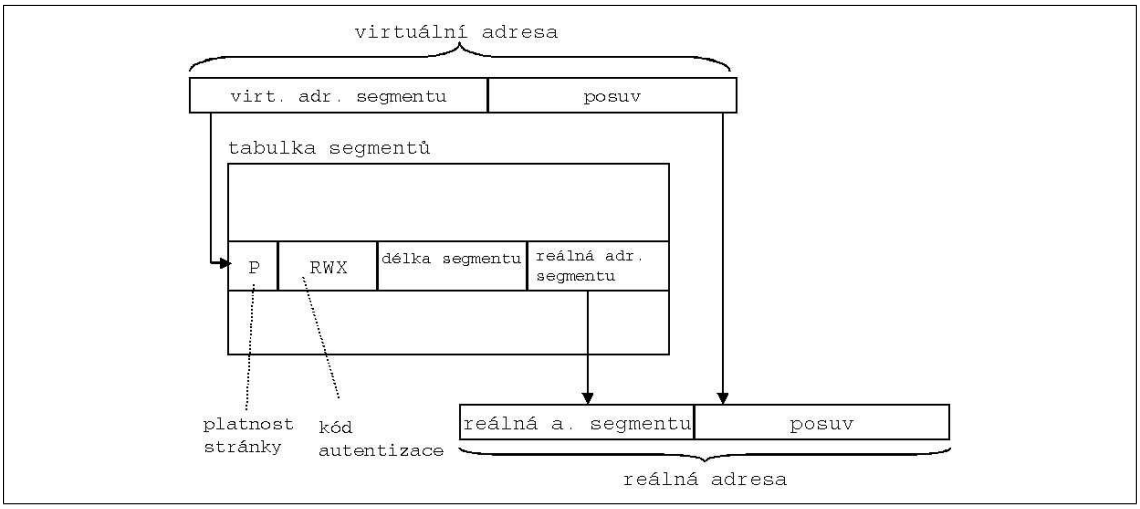
Tato paměť umožňuje dosáhnout úspory kapacity HP tím, že se program do HP zavádí po částech. *Segment* je skupina po sobě následujících paměťových míst, která mohou měnit svou velikost. *Segment* vždy zaujímá v HP souvislé místo. Adresy v *segmentu* jsou relativní vůči začátku *segmentu* ⇒ *segmenty* mohou být v HP přemístitelné. Logická adresa se skládá z obsahu segmentového registru a posunutí - viz. obrázek 5.9.




Obrázek 5.7: dvoúrovňová organizace




Obrázek 5.8: stránkovaná segmentovaná paměť




Obrázek 5.9: segmentovaná paměť