# 6. Paměťový systém počítače

Paměť je zařízení pro uchování dat a programů (nejen v počítači). Ve von Neumannově architektuře jsou data i instrukce v „jedné“ paměti, ale reálně existuje struktura různých typů pamětí různých technologií tak, aby uživatel získal dojem mnohem větší kapacity paměti, kterou může využívat, a mnohem větší rychlosti (tedy přístupu k datům). K tomu slouží paměťová hierarchie, viz obrázek 6.1. Základními funkcemi pamětí je zápis do paměti a čtení z paměti. Existují různé typy pamětí, které se dělí podle různých kritérií:

- použití v počítači (hlavní paměť, skrytá paměť, vnější paměť, ....),
- fyzikálního principu (polovodičová paměť, magnetická paměť, optická paměť, ...),
- způsobu výběru položek (paměť s adresovým výběrem, paměť s postupným výběrem, asociativní paměť, zásobník, fronta, ...),
- způsobů a možností změny uložené informace (RWM, RAM, ROM, PROM, EPROM, EEPROM, FLASH, ...),
- vztahu k napájení: volatilní (uložená informace zanikne po vypnutí napájení) a nonvolatilní (obsah zůstane uchován i po vypnutí napájení).

| Typ paměti | Typická realizace | Doba přístupu | Kapacita |
|---|---|---|---|
| Registry | Klopné obvody | jednotky ns | Desítky-stovky bytů |
| Cache (skrytá paměť) | Statická RAM | ~ 10 ns | ~ MB ... GB |
| Hlavní paměť (operační paměť) | Dynamická RAM | ~ 50ns | ~ MB ... GB |
| Vnější paměť | Magnetický disk „hard“ disk | ~ 10 ms | ~ GB |
| Záložní paměť | CD, DVD magnetická páska flash disk | ~ stovky ms ... s | ~ GB ... TB |

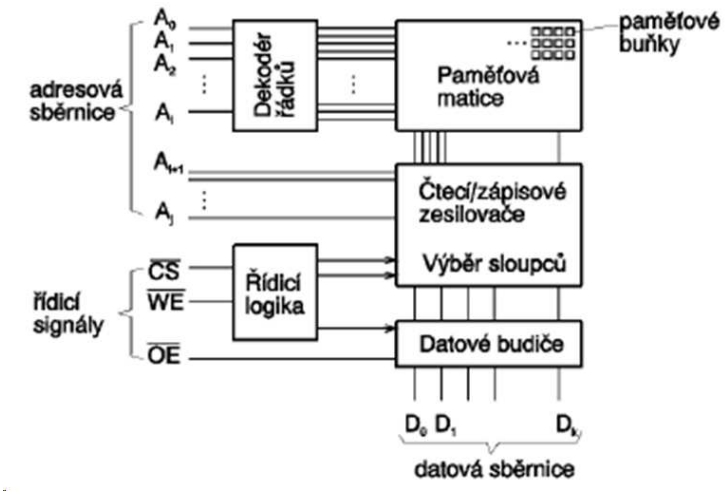
Obrázek 6.1.: Paměťová hierarchie.

Základní terminologie u pamětí v počítači je následující:

- paměťová buňka ... základní stavební blok paměti, slouží k záznamu jednoho bitu,

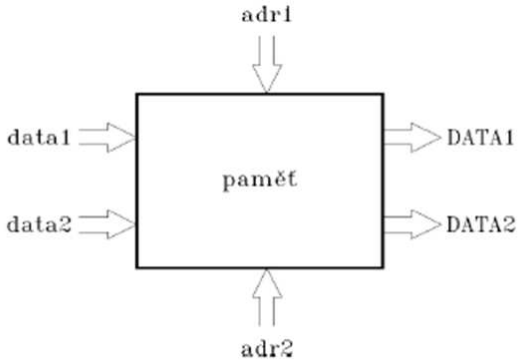
- paměťové místo ... skupina paměťových buněk, do kterých lze najednou zapisovat nebo je číst (šířka slova paměti),
- položka ... obsah paměťového místa,
- adresa ... číselné označení (index) paměťového místa, jímž lze vybírat jednotlivé položky,
- kapacita paměti ... maximální počet položek, které lze do paměti uložit,
- paměťová matice ... skupina paměťových míst uspořádaná tak, že je lze vybírat adresou.

Typický adresovatelný paměťový obvod (viz obrázek 6.2) obsahuje kromě paměťové matice, dekodéry řádků a sloupců (typicky dekodéry z binárního kódu do kódu 1 z N, z důvodů šetření počtu vstupních pinů), řídící logiku pro zpracování signálů pro zápis ($\overline{WE}$: write enable ... povolení zápisu), výběr obvodu ($\overline{CS}$: chip select ... výběr čipu, podmiňuje provedení zápisu nebo čtení) a čtení z paměti ($\overline{OE}$: output enable ... aktivace výstupních třístavových budičů datové sběrnice). Tyto signály bývají typicky aktivní v nule, proto se označují s negací.




Obrázek 6.2.: Typická organizace paměťového obvodu.

Paměti mohou být přizpůsobeny potřebám aplikace, používají specializované paměťové čipy. Existuje i tzv. vícebránová paměť, kde do paměťové matice (paměťového obvodu) je možné přistupovat na dvě či více různých míst pomocí více adresových sběrnic (viz obrázek 6.3). Takový typ paměti je efektivní využít pro realizaci rychlých pamětí typu fronta. Ve složitějších programovatelných obvodech (FPGA) je možné využít specializované bloky pamětí nebo si navrhnout vlastní organizaci paměti potřebné kapacity a uspořádání. Dále si na příkladech ukážeme, jak připojit paměť k procesoru, jak zvýšit kapacitu paměti, pokud má procesor širší adresovou sběrnici, jak připojit více bloků paměti a jak rozšířit šířku dat, která z hlavní paměti získáme najednou.




Obrázek 6.3.: Dvoubránová paměť.

## 6.1. Příklady

1. Kolik vstupů a výstupů bude mít paměťový blok adresovatelné paměti s kapacitou 16 x 4, 256 x 8, 1K x 9 bitů?

   **Řešení**
   Kapacita paměti je dána velikostí paměťové matice a potřebnou velikostí dekodéru, tedy pro adresaci paměťové matice o velikosti 16 x 4 potřebujeme 4 adresové vstupy a 4 výstupy, pro 256 x 8 8 vstupů a 8 výstupů a pro 1K x 9 10 vstupů a 9 výstupů.

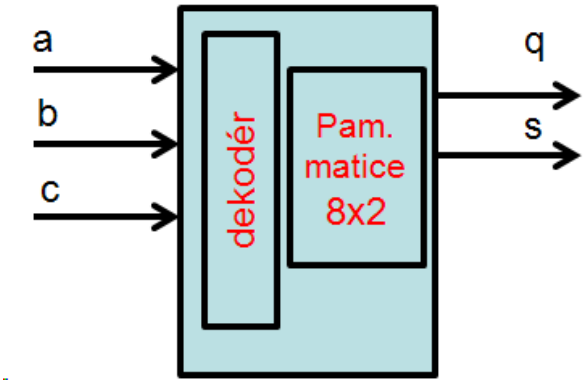
2. Navrhněte jeden blok binární sčítačky pomocí paměti vhodné velikosti.

   **Řešení**

| a | b | p | q | s |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Obrázek 6.4.: Tabulka pro binární sčítačku.

Tabulka pro jeden blok binární sčítačky je na obrázku 6.4. Odtud vyplývá, že potřebujeme paměťovou matici, do které se vejde tato tabulka, tedy 8 x 2. Výsledné schéma (vstupy $\overline{CS}$, $\overline{OE}$ a $\overline{WE}$ si domyslete) je na obrázku 6.5. Na rozdíl od realizace pomocí hradel (kapitola 2) při realizaci kombinačního obvodu pomocí adre-

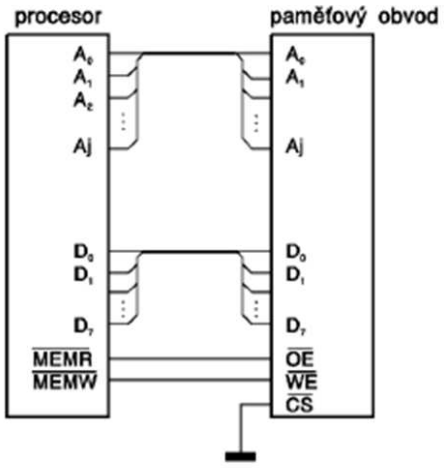



Obrázek 6.5.: Binární sčítačka realizovaná pamětí.

sovatelné paměti, nemůžeme minimalizovat, ale musíme použít celou pravdivostní tabulku funkce (jinak řečeno úplnou normální formu funkce). Důvodem je přístup do paměťové matice přes adresový dekodér, a tedy úplnou adresu.

3. Realizujte připojení bloku hlavní paměti k procesoru. Uvažujte:
a) stejnou šířku adresové sběrnice,
b) modul hlavní paměti o kapacitě 96 KB slabikově organizovanou (96 K x 8) realizovanou z modulů o kapacitě 32 K x 8 (32 KB),
c) modul hlavní paměti o kapacitě 32 K x 16 z modulů o kapacitě 32 K x 8, tedy paměť adresovatelnou po slovech.

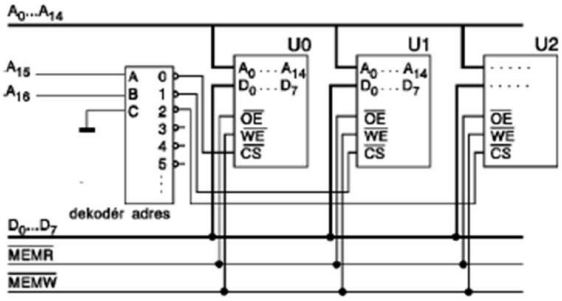
**Řešení**
a) nejjednodušší je přímé propojení obou obvodů podle obrázku 6.6.




Obrázek 6.6.: Přímé připojení paměti k procesoru.

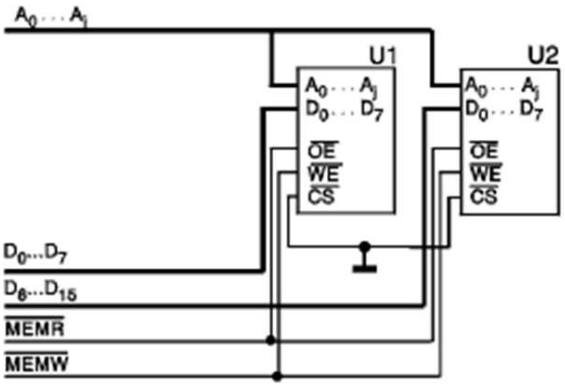
b) Budeme potřebovat 3 moduly, každý z modulů o velikosti 32KB je adresován 15 bity, vyšší bity adres jsou použity pro výběr jednoho ze tří modulů. Potřebujeme tedy ještě vnější dekodér adres - dekodér z binárního kódu do kódu 1 z N, připojený na řídící vstupy $\overline{CS}$. Pro náš případ by stačil i menší, než je na obrázku 6.7. Tento

dekodér adres by mohl vybírat z $2^3$ tedy z 8 paměťových obvodů-bloků pomocí 18 bitových adres, paměť by se rozšířila na 256KB.




Obrázek 6.7.: Rozšíření kapacity hlavní paměti.

c) Rozšíření šířky dat je jednodušší, protože vybíráme najednou z obou bloků paměti podle obrázku 6.8.




Obrázek 6.8.: Rozšíření slova hlavní paměti.

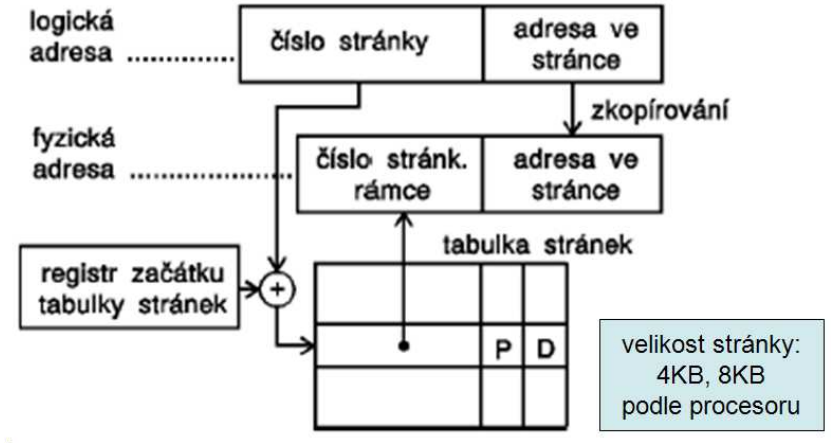
## 6.2. Virtualizace paměti: stránkování a segmentace

Virtualizace paměti je systém několika pamětí s různými parametry (kapacita, rychlost) řízený tak, aby vytvářel paměťové prostory potřebné velikosti pro program a data. Umožňuje realizaci jednoho nebo několika logických (virtuálních) adresových prostorů, kde každý může být větší než skutečná kapacita hlavní paměti. Prostoru pro programy a data v hlavní paměti říkáme fyzický paměťový prostor, logické adresové prostory jsou ve skutečnosti ve vnější paměti (na disku). Části programů a data jsou přesouvány do hlavní paměti, požaduje-li k nim procesor přístup. V hlavní paměti jsou tedy jen ty programy a data, se kterými procesor právě pracuje. To má svoje nesporné výhody: lze přemístit části programu bez nutnosti je znovu překládat, zajistit ochranu dat před neoprávněným přístupem a nežádoucí modifikací díky tomu, že pracujeme s logickými adresami a ne s absolutními. Na druhé straně musíme správným a rychlým způsobem zajistit přesuny dat mezi různými paměťmi v rámci paměťové hierarchie a zároveň dodržet

konzistenci dat tak, aby procesor měl k dispozici aktuální a správná data. Hlavní paměť se adresuje fyzickými adresami, překlad logických adres na fyzické zajišťuje mechanismus virtuální paměti. Tyto mechanismy jsou dva, a to **stránkování** a **segmentace**.

**Stránkování** je založeno na principu rozdělení logického adresového prostoru na úseky pevné délky, které se nazývají stránky (logické stránky), a rozdělení fyzického adresového prostoru na stejně velké úseky, kterým říkáme stránkové rámce (fyzické stránky). Logický adresový prostor je realizován ve vnější paměti. Data (úseky programu) se přesouvají do hlavní paměti po jednotlivých stránkách, jsou-li v průběhu výpočtu požadována a pokud se příslušná stránka již v paměti nenachází. Překlad (určení, do kterého stránkového rámce v hlavní paměti se stránka přesune) používá datovou strukturu nazvanou **tabulka stránek**. Tabulka stránek je uložena v hlavní paměti, obsahuje pro každou logickou stránku jednu položku a tato položka obsahuje informaci, zda se daná stránka nachází v hlavní paměti. Pokud ano, tak kde (v kterém stránkovém rámci). Tento stránkovací mechanismus je znázorněn na obrázku 6.9. V tabulce stránek je uložena horní část fyzické adresy, příznak přítomnosti stránky v hlavní paměti, příznak změny dat ve stránce (zda bylo po dobu přítomnosti v HP do stránky zapisováno, tzv. Dirty bit) a další bity, které určují, zda je vhodné stránku přepsat (vyhodit z hlavní paměti např. podle toho, kdy byla použita).

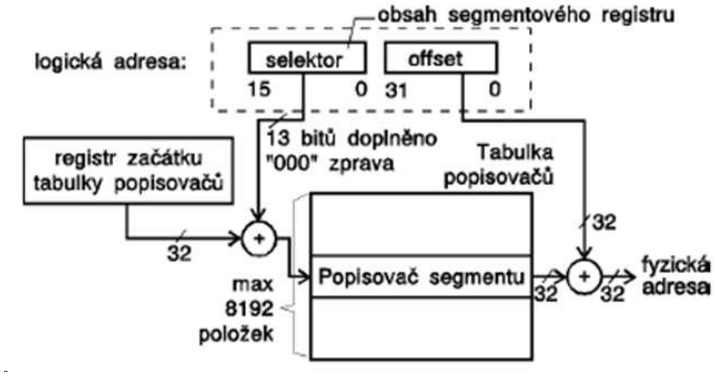
Je-li stránka přítomna v hlavní paměti, přeloží se logická adresa na fyzickou (pozná se to podle příznaku přítomnosti stránky v HP). Není-li stránka přítomna, vyvolá se přerušení. Přerušovací mechanismus realizuje načtení stránky z vnější paměti. Jestliže potřebujeme logickou stránku, která není v hlavní paměti, a zároveň je nutné některou stránku v hlavní paměti přepsat (není volný žádný stránkový rámec), musíme ji předtím zkopírovat do logického paměťového prostoru, ale jen v případě, že do stránky bylo zapisováno. To poznáme pomocí Dirty bitu. Pokud nebyla data měněna, lze stránkový rámec jen přepsat a nikam nekopírovat, čímž se ušetří čas. Strategie výběru toho, kterou stránku přepsat, a jak tento mechanismus realizovat přesahuje obsah předmětu BI-SAP (dozvíte se to v BI-JPO a v BI-APS).




Obrázek 6.9.: Princip stránkování.

**Segmentace** vychází z toho, že programy mají různou velikost, „stránka“ může být zbytečně velká nebo naopak malá. Dělíme tedy paměťový prostor na segmenty, funkčně

samostatné části programu proměnné délky, které lze do hlavní paměti zavádět v případě potřeby. Adresy v segmentu jsou relativní vůči začátku (tzv. bázi) segmentu, což umožňuje přemístitelnost segmentů v hlavní paměti. Logická adresa se skládá z báze segmentu a offsetu (posunutí). Báze segmentu je uložena buď v segmentovém registru nebo v tabulce popisovačů segmentů. Segmentové registry byly použity v procesorech Intel 8086, které měly jen 16bitové registry a potřebná 20bitová adresa do hlavní paměti byla tvořena kombinací segmentového registru a „relativního programového čítače". Tak byl umožněn masivní rozvoj osobních počítačů, tedy adresováním a rozšířením hlavní paměti na velikost 1 MB. Princip použití tabulek popisovačů segmentů je na obrázku 6.10. Tato tabulka obsahuje na rozdíl od tabulky stránek nejen adresu začátku segmentu, tzv. bázi segmentu, ale i jeho velikost. V řešení podle obrázku 6.10 je ještě bit granularity, který určuje, zda je tato velikost vyjádřena ve slabikách nebo v blocích o velikosti 4 KB. To umožnilo adresovat fyzický adresový prostor až 4 GB a logický adresový prostor $2^{13}$ segmentů o velikosti až 4 GB a přiřadit segmentům přístupová práva.




Obrázek 6.10.: Princip segmentace v procesoru Intel 80386 v tzv. „chráněném režimu".

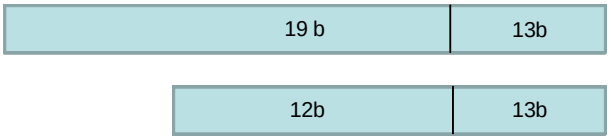
### 6.2.1. Příklady

1. Hlavní paměť má velikost 32 MB (25 bitová adresa), vnější paměť 4 GB (32 bitová adresa) a velikost stránky je 8 KB. Jak velká bude tabulka stránek a jak velkou část hlavní paměti zabere?

   **Řešení**
   Vyjdeme z rozdělení adresy na adresu ve stránce a zbytek, viz obrázek 6.11. Adresa ve stránce o velikosti 8 KB má šířku 13 bitů. Tabulka stránek musí mít $2^{19}$ položek, každá položka musí obsahovat 12 bitů pro adresu stránkového rámce a další příznaky (Dirty bit, bit platnosti, ....) tedy 2 B. Velikost tabulky stránek $2^{20} = 1$ MB což je 1/32 velikosti hlavní paměti, tedy asi 3%.

2. Hlavní paměť má velikost 4 MB. Vnější paměť 4 GB a velikost stránky 4 KB. Jak velká bude tabulka stránek a jak velkou část hlavní paměti zabere?

   **Řešení**
   Vyjdeme z příkladu 1. Adresa ve stránce o velikosti 4 KB má šířku 12 bitů, adresa

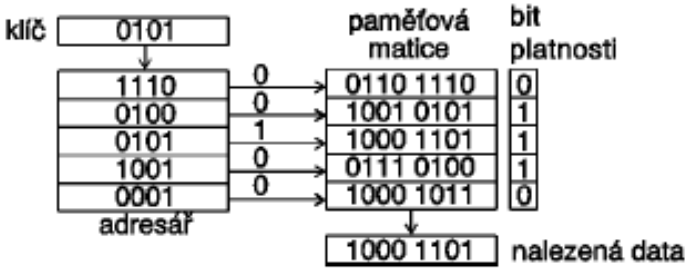



Obrázek 6.11.: Příklad rozdělení adresy.

v hlavní paměti 22 bitů a ve vnější paměti 32 bitů. Potom je potřeba, aby měla tabulka stránek $2^{32-12} = 2^{20}$ položek, každá položka $22 - 12 = 10$ bitů pro adresu ve stránkovém rámci, tedy 2 B. Velikost tabulky stránek $2^{20}$ x 2 B = 2 MB, což je polovina velikosti hlavní paměti, takže toto řešení není použitelné. Pro velký poměr velikosti hlavní a vnější paměti je nutné volit jiné řešení než stránkování, například dvouúrovňovou tabulku stránek nebo kombinaci se segmentováním.

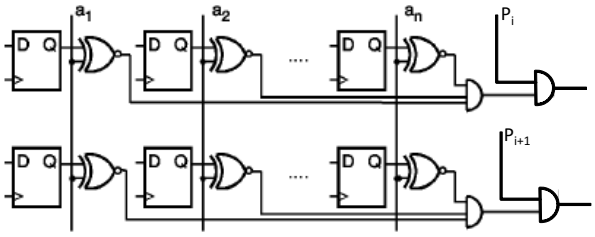
## 6.3. Skrytá paměť: cache

Principem skryté paměti (cache, ale někdy též buffer memory, rychlá vyrovnávací paměť) je využití asociativního přístupu k položkám, tedy ne přístupu pomocí adresy, ale pomocí obsahu (přesněji řečeno části obsahu). Je to „malá“ a rychlá paměť zařazená mezi procesor a hlavní paměť, která obsahuje kopie nejčastěji používaných položek hlavní paměti. Tyto skryté „cache“ paměti se využívají i pro zrychlení hledání v tabulce stránek nebo v řadičích vnějších pamětí na disku. Adresuje se částí datové položky, která se má vyhledat, tzv. klíčem. Na rozdíl od adresovatelné paměti zde není adresový dekodér, ale **adresář**. Nejprve si vysvětlíme princip tzv. plně asociativní paměti, potom asociativní paměti s omezeným stupněm asociativity, který umožní použít paměť typu SRAM jako cache. U plně asociativní paměti se hledaný klíč srovná najednou se všemi položkami v adresáři, viz obrázek 6.12. Každá položka adresáře obsahuje logické obvody umožňující najednou prohledat všechny položky adresáře podle klíče. Při shodě se přečte/zapíše položka do paměťové matice, která odpovídá pozici nalezeného klíče v adresáři, viz obrázky 6.12 a 6.13.



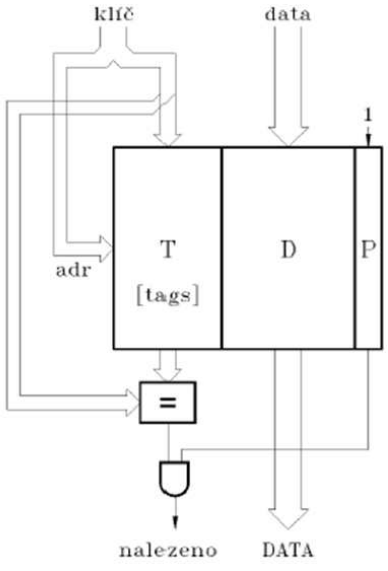

Obrázek 6.12.: Princip plně asociativní paměti s kapacitou 5B.

Nyní si vysvětlíme jak a proč se skrytá paměť (cache) v počítači používá s tím, že se nebudeme zabývat víceúrovňovými typy ani strategiemi, které položky a jak do cache ukládat a vyhazovat. Základem je princip tzv. lokality dat (časová lokalita: když procesor



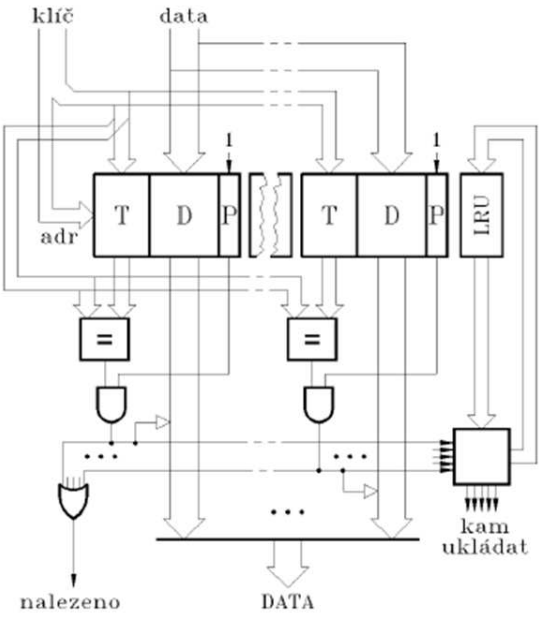

Obrázek 6.13.: Realizace adresáře plně asociativní paměti.

používá nějakou položku v paměti, je vysoká pravděpodobnost, že ji bude používat znovu nebo prostorová lokalita: že procesor bude používat položky v paměti umístěné poblíž právě používané). Data v cache jsou proto kopie často používaných položek dat z hlavní paměti, klíčem je adresa položky (přesnější část adresy položky podle typu cache). Velmi důležitý je bit platnosti P, viz obrázek 6.13. Při čtení z paměti se zahájí cyklus čtení současně z cache i z hlavní paměti. Pokud se položka v cache nalezne, cyklus hlavní paměti se nedokončí. V opačném případě se přečtou data z hlavní paměti (a obvykle zároveň uloží do cache). Při zápisu dat pokud položka v cache není, zapíše se zpravidla jen do hlavní paměti. Pokud v cache je, je možné postupovat dvěma způsoby: nová hodnota se zapíše zároveň do cache i do hlavní paměti (průběžný zápis: *write through*). Nebo se nová hodnota zapíše jen do cache, ale při uvolňování položky z cache se musí její obsah přepsat do hlavní paměti, pokud byla modifikována po dobu přítomnosti v cache (odložený zápis: *write back*). Z tohoto popisu je vidět, že cache je pro programátora zcela průhledná. Podle toho je třeba navrhnout příslušný hardware, aby procesor pracoval mnohem rychleji, než při realizaci cyklu zápis-čtení do hlavní paměti. Skrytá paměť bývá umístěna přímo na čipu (alespoň tzv. L1 cache).




Obrázek 6.14.: Realizace asociativní paměti se stupněm asociativity 1 (přímo mapovaná cache).

Nevýhodou plně asociativní paměti je, že adresář je tvořen speciálními obvody, takže




Obrázek 6.15.: Realizace asociativní paměti se stupněm asociativity 2.

při stejné kapacitě zabere tato paměť trojnásobnou plochu čipu. Řešením je omezený stupeň asociativity, tzn. že každé položce je určeno místo (nebo několik míst), kde se může nacházet. Toto místo je určené částí adresy položky. Klíč z obrázku 6.12 je rozdělen na dvě části, na tu část adresy (zde označené *adr*), podle které se určí místo, kde se hledaná položka může vyskytovat a na „podklíč“ (zde označený jako *T, Tags*). Potom je adresář možné realizovat běžnou pamětí RAM. Přítomnost položky se zjistí porovnáním s podklíčem (nebo několika) uloženými v adresáři (v bloku *T*). Stupeň asociativity je počet míst, na kterých se položka může nacházet. Pro zvýšení efektivity jsou data v oblasti *D* uložena po blocích (řádcích) o velikosti např. 16 nebo 32 slabik.

Kapacita cache je jako u každé paměti určena velikostí paměťové matice, tedy dat, které je možné do ní uložit. Pro realizaci je ale třeba tak velká adresovatelná paměť, aby se do ní vešel i adresář, tedy blok *T*, protože i část klíče je uložena do bloku paměti, tvoří adresář, zatímco druhá část adresy (*adr*) slouží k adresovému vyhledávání. Princip pro stupeň asociativity 1, které se říká **přímo mapovaná** cache (*direct mapped*), je na obrázku 6.14. V tomto případě je možné data se stejnou částí adresy (na obrázku označena jako *adr*), uložit jen na jedno místo – řádek v paměťové matici. Zbylá část adresy – klíče (*T*), uložená do adresáře, se může měnit. Celá sdresa je rozdělena na 3 části: podklíč (*T*), adresu řádku (*adr*) a adresu slabiky v bloku. TAG je tvořen nejvyššími bity adresy, aby se mohly v cache vyskytovat položky s „blízkými“ adresami (aby se nemusel obsah cache často přepisovat, protože obvykle se pracuje s daty uloženými blízko sebe, viz princip lokality). Pokud bude stupeň asociativity vyšší (2, 4, ...), počet bloků se zdvojí či zečtyřnásobí, viz obrázek 6.15. Blok LRU (ve významu „nejdéle nepoužito“) slouží k výběru ze 2, 4, ... míst, na které se mají data uložit. Podrobněji v předmětech BI-JPO a BI-APS.

### 6.3.1. Příklady

1. Jaký stupeň asociativity má plně asociativní paměť?

   **Řešení**
   Každá položka může být uložena na libovolném místě paměťové matice, její adresa se získá srovnáním klíče a obsahu adresáře, takže správná odpověď je, že stupeň asociativity plně asociativní paměti je roven její kapacitě.

2. Z kolika možností se vybírá (např. pomocí LRU) při přepisování (vyhazování) bloku při stupni asociativity 1, 2, 4?

   **Řešení**
   Stupeň asociativity určuje počet míst, na které se může položka se stejnou částí adresy uložit, takže správná odpověď je, že z tolika míst, kolik je stupeň asociativity.

3. Kapacita hlavní paměti je 64 KB, kapacita cache je 32 B, velikost bloku 4 B, stupeň asociativity je 2, konkrétní obsahy hlavní paměti viz obrázek 6.16 a cache viz obrázek 6.17. Co je uloženo na na adrese 0FFE? Je tato položka i ve skryté paměti (cache)?

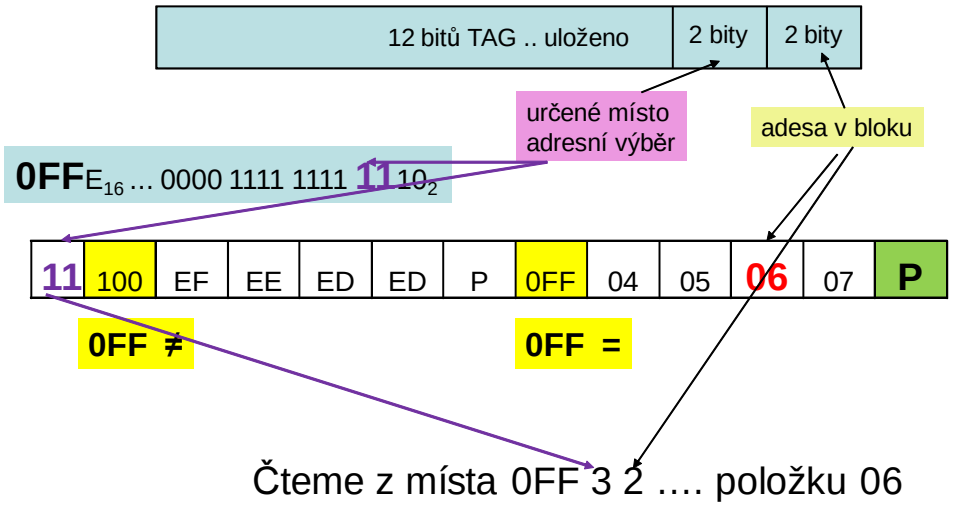
| adresa | | položky | | |
|---|---|---|---|---|
| .... | | | | |
| 0FF0 | F0 F1 F2 F3 | F4 F5 F6 F7 | 00 01 02 03 | 04 05 **06** 07 |
| 1000 | 08 09 0A 0B | 0C 0D 0E 0F | FF FE FD FC | EF EE ED ED |
| ..... | | | | |

Obrázek 6.16.: Část obsahu HP z příkladu 3.

| adresa | TAG | Blok dat | | | | platnost | TAG | Blok dat | | | | Platnost |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 987 | 10 | 20 | 30 | 40 | P | 100 | 08 | 09 | 0A | 0B | P |
| 1 | 100 | 0C | 0D | 0E | 0F | P | 0FF | 0C | 0D | 0E | 0F | N |
| 2 | 000 | 11 | 22 | 33 | 44 | N | 321 | F1 | F2 | F3 | F4 | P |
| 3 | 100 | EF | EE | ED | ED | P | 0FF | 04 | 05 | **06** | 07 | P |

Obrázek 6.17.: Obsah cache z příkladu 3.

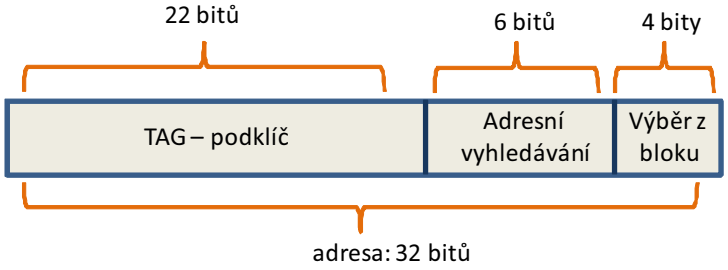
**Řešení**
Obsah adresy 0FFE je 06, jak je vidět z výpisu hlavní paměti. Podle velikosti cache, velikosti bloku a stupně asociativity vychází, že cache je uspořádaná do dvou bloků po 16 slabikách, tedy podle obrázku 6.17. Rozdělení adresy a způsob hledání je znázorněno na obrázku 6.18. Pro danou adresu se data mohou vyskytovat pouze v posledním řádku (část adresy 11) a hledáme, zda je v něm ve zvýrazněném sloupci „TAG“ uloženo 0FF. Odpověď je tedy, že obsah adresy 0FFE je 06 a data jsou v cache uložena.




Obrázek 6.18.: Řešení příkladu 3.

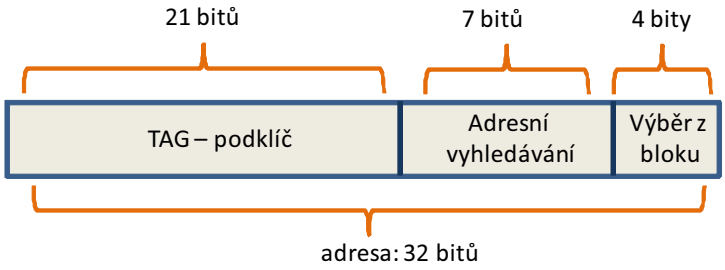
4. Kolik potřebujete srovnávacích obvodů (ekvivalencí) pro procesor s 32bitovou fyzickou adresou, skrytou pamětí (cache) o velikosti 4 KB, stupněm asociativity 4, kde do cache ukládáte bloky dat o velikosti 16 B (slabik)? Které bity adresy budou tvořit podklíč ($T$, TAG)? Nakreslete rozdělení adresy. Jak se situace změní pro stupeň asociativity 2?

   **Řešení**
   Nejnižší část adresy je adresa v bloku, tedy pro výběr jedné slabiky ze 16 slabik potřebujeme 4 bity adresy. Cache o velikosti 4 KB se stupněm asociativity 4 bude organizovaná do 4 částí po 1 KB. Pro adresaci 1 KB potřebujeme 10 bitů, ale vyhledáváme bloky po 16B, tedy $10 - 4 = 6$. Část adresy pro adresní vyhledávání *adr*) bude 6 bitová. Zbylé bity (tedy $32 - 6 - 4 = 22$) budou uloženy v adresáři cache pro asociativní vyhledávání $T$ (TAG-podklíč). A protože máme cache se stupněm asociativity 4, budeme potřebovat 88 srovnávacích obvodů ($4 \times 22$). Situace je znázorněna na obrázku 6.19. Jestliže bude stupeň asociativity 2 pro stejnou velikost hlavní i skryté paměti, změní se rozdělení adresy podle obrázku 6.20.




Obrázek 6.19.: Rozdělení adresy z příkladu 4 pro s=4.

5. Jak velký bude TAG (T, část adresy, podklíč), který budete ukládat do adresáře skryté paměti (cache) a podle kterého budete asociativně vyhledávat pro procesor s 24bitovou fyzickou adresou, skrytou pamětí (cache) o velikosti 4 KB, stupněm asociativity 4 (2), kde do cache ukládáte bloky dat o velikosti 8 B (slabik)? Nakreslete

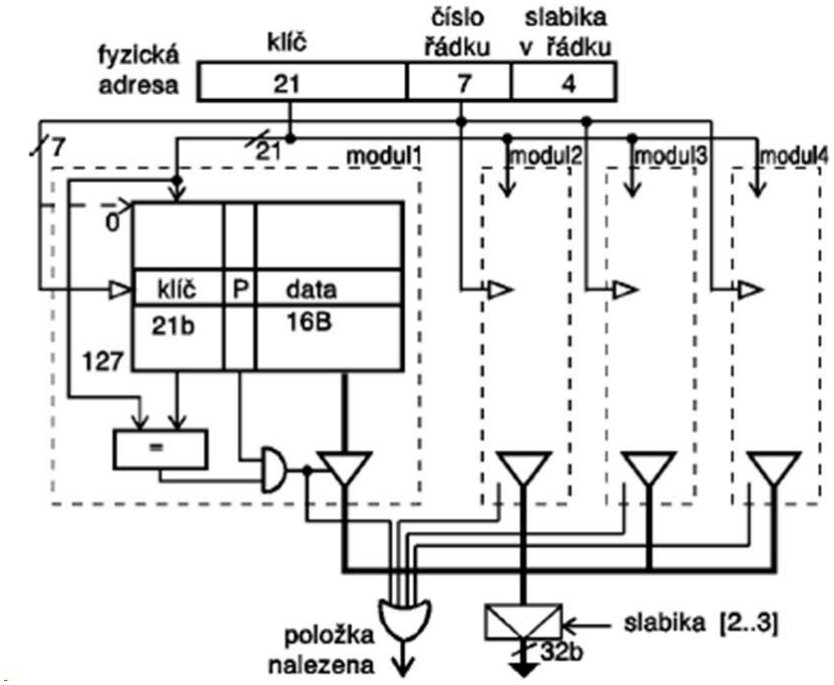



Obrázek 6.20.: Rozdělení adresy z příkladu 4 pro s = 2.

obrázek.

6. Jak široká část klíče bude použita pro adresní vyhledávání (*adr*), tzn. číslo řádku položek ve skryté paměti (cache) v procesoru s 30bitovou fyzickou adresou, skryté pamětí (cache) o velikosti 8 KB, stupněm asociativity 2, kde do cache ukládáte bloky dat o velikosti 16 B (slabik)?

7. Jak se změní rozdělení fyzické adresy a struktura cache z obrázku 6.21, jestliže bude stupeň asociativity 2 (8)?




Obrázek 6.21.: Obsah cache z příkladu 7.

**Řešení**

Pro stupeň asociativity 2 budou třeba jen 2 moduly, ale pro stejnou kapacitu cache budou mít moduly dvojnásobnou velikost. Pro stejně velké bloky bude tedy třeba 8 bitů pro adresní vyhledávání (*adr*)) a tudíž do adresáře bude uložen jen 20bitový podklíč (TAG). Naopak, pro stupeň asociativity 8, bude cache organizována do 8 modulů po 1 KB, tedy bude třeba jen 6 bitů pro adresní vyhledávání, naopak TAG

bude 22 bitů. Všimněte si, že se zvyšováním stupně asociativity se zvětšuje adresář a je mnohem složitější výběr přepisovaného bloku. Tedy hardware je složitější.

8. Mějme paměť cache o velikosti 64 B, stupněm asociativity 2, velikostí bloku 4 B a hlavní paměť s kapacitou 128 KB slabikově organizovanou.
a) nalezněte v cache (viz obrázek 6.22, vše je šestnáctkově) data s adresou: 00110010000111001
b) Uložte do cache (přepište všechna příslušná pole v obrázku) blok dat AA, BB, CC, DD od adresy: 00001111101001000

| adresa | TAG | data | | | | platnost | TAG | data | | | | platnost |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 987 | 11 | 21 | 3A | 4A | P | 100 | 08 | 09 | 0A | 0B | P |
| 1 | 100 | 0C | 0D | 0E | 0F | P | 0FF | 0C | 0D | 0E | 0F | N |
| 2 | 000 | 11 | 22 | 33 | 44 | N | 32B | F1 | F2 | F3 | F4 | P |
| 3 | 100 | FF | 0F | FD | ED | P | 0FF | 04 | 05 | 06 | 07 | P |
| 4 | 987 | 10 | 20 | 30 | 40 | P | 100 | 08 | 09 | 0A | 0B | P |
| 5 | 100 | 0C | 0D | 0E | 0F | P | 0FF | 0C | 0D | 0E | 0F | N |
| 6 | 321 | 11 | 22 | 33 | 44 | N | 321 | F1 | **F2** | F3 | F4 | P |
| 7 | 100 | EF | EE | ED | ED | P | 0FF | 04 | 05 | 06 | 07 | P |

Obrázek 6.22.: Obsah cache z příkladu 8a).

**Řešení**
a) Pro výběr slabiky v bloku jsou určeny nejnižší dva bity (zde 01, vybíráme tedy z druhého sloupce dat). Pro adresu řádku, který vybíráme z osmi možných podle kapacity cache, jsou určeny další tři bity (zde 110). Hledáme, kde je v adresáři uložena část adresy 001100100001 (321) a zároveň nastaven bit platnosti. Hledaná data jsou F2.

b) Rozdělení adresy je 0000 (0) 1111 (F) 1010 (A) řádek 010 (adresa 2) a 00. Přepsaná data jsou zvýrazněna na obrázku 6.23. Pokud by byly nastaveny bity platnosti pro oba bloky dat, musela by být uplatněna nějaká strategie výběru přepisované položky. Implementací těchto strategií se tento předmět nezabývá.

| adresa | TAG | data | | | | platnost | TAG | data | | | | platnost |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 987 | 11 | 21 | 3A | 4A | P | 100 | 08 | 09 | 0A | 0B | P |
| 1 | 100 | 0C | 0D | 0E | 0F | P | 0FF | 0C | 0D | 0E | 0F | N |
| 2 | **0FA** | **AA** | **BB** | **CC** | **DD** | **P** | 32B | F1 | F2 | F3 | F4 | P |
| 3 | 100 | FF | 0F | FD | ED | P | 0FF | 04 | 05 | 06 | 07 | P |
| 4 | 987 | 10 | 20 | 30 | 40 | P | 100 | 08 | 09 | 0A | 0B | P |
| 5 | 100 | 0C | 0D | 0E | 0F | P | 0FF | 0C | 0D | 0E | 0F | N |
| 6 | 321 | 11 | 22 | 33 | 44 | N | 321 | F1 | F2 | F3 | F4 | P |
| 7 | 100 | EF | EE | ED | ED | P | 0FF | 04 | 05 | 06 | 07 | P |

Obrázek 6.23.: Řešení příkladu 8b.