Virtuální paměť

Nová látka: Virtuální paměť (VM)

Pozorování: *Cache vyrovnává rychlosti procesoru a hlavní paměťi*

…Hlavní paměť je „cache“ pro diskovou paměť…

Upřesnění:

- VM umožňuje efektivní a bezpečné sdílení paměti mezi více programy (podpora multiprogramového zpracování)
- VM odstraňuje potíže s malým rozsahem fyzické paměti
- VM zjednodušuje *zavádění* programů podporou *relokace*

Historie: Nejdříve byla vyvinuta VM, teprve pak cache.

Cache je založena na technologii VM, ne naopak.

Virtuální paměť - pojmy

Stránka (Page): *Blok* ve virtuální paměti (cache = blok, VM = page)

Výpadek stránky: Neúspěch při operaci MemRead
(cache = miss, VM = výpadek stránky)

Fyzická adresa: Adresa mířící do fyzické paměti

Virtuální adresa: Určena CPU, odpovídá *velkému* adresnímu prostoru, je transformována pomocí HW + SW na fyzickou adresu

Mapování paměti nebo **translace adresy:**

Proces transformace virtuální adresy na fyzickou adresu

Translation Lookaside Buffer (TLB):

Zvyšuje efektivitu procesu transformace adresy

Fyzická vs. virtuální paměť

- **Fyzická paměť**
  - *Instalována v počítači:* ~ 512MB – 4 GB RAM v PC
  - Limitována velikostí a spotřebou
  - *Potenciální rozšíření limitováno velikostí adresního prostoru*
- **Virtuální paměť**
  - **Jak uložit 2^{32} *slov* do vašeho PC?**
  - **Nikoliv jako RAM – nedostatek prostoru nebo napájení**
  - Nechť CPU „věří“ , že má k dispozici 2^{32} slov
  - Hlavní paměť pak pracuje jako pomalejší cache
  - *Stránka* - jednotka pro přesun obsahu paměti na/z disk(u)
  - Jednotka Memory Management Unit využívá tabulku stránek (adresovací systém) při řízení přesunu stránek z/do hlavní paměti

Motivace

- Předpokládejme soubor programů, běžících na počítači současně.
  - Celková kapacita paměti, kterou tyto programy potřebují může být mnohemmvětší, než je celá kapacita hlavní paměti stroje.
  - V daném okamžiku se využívá jenom malý zlomek celé kapacity hlavní paměti.
  - Hlavní paměť musí obsahovat pouze aktivní části všech programů – působí vlastně jako cache.
  - Aby takový systém pracoval úspěšně, musíme zaručit, aby každý program prováděl čtení/zápis jen v oblasti, která přísluší právě jemu (separace).

Virtuální paměti a adresní prostor

- Jakmile je program přeložen, nevíme jaké programy budou zpracovávány současně (navíc se toto může při každém spuštění měnit).
- Proto je každý program překládán jakoby pro práci ve svém vlastním *adresním prostoru*. To je oddělený úsek paměti, dostupné jen tímto programem.
- *Systém virtuální paměti* (VM) počítače implementuje translaci z adresního prostoru programu do fyzického adresního prostoru.

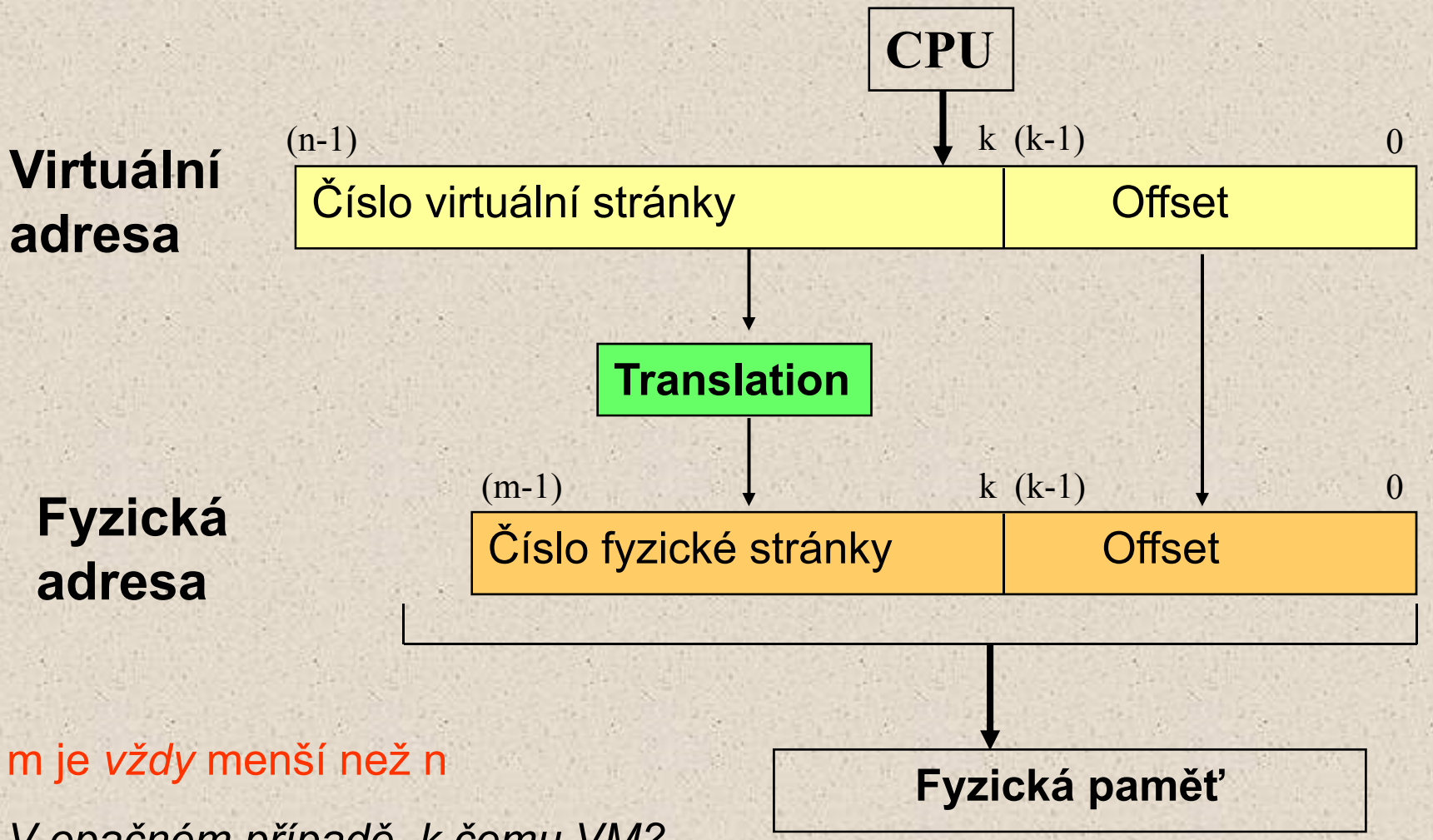
Další motivace

- Další motivací pro virtuální paměť je možnost, aby i jediný program mohl překročit svou velikostí kapacitu fyzické paměti.
- Paměť, která se nepoužívá, může být přenesena na disk a opět načtena zpět, pokud systém virtuální paměti rozhodne (podobně jako je tomu mezi cache a hlavní pamětí).
- Virtuální paměť automaticky spravuje dvě úrovně hierarchie paměti, reprezentované hlavní/fyzickou pamětí a sekundární pamětí (obvykle disk).

Overlaye

- Před příchodem mechanizmu virtuální paměti (~ 1960 Manchester) museli programátoři přímo organizovat využívání sekundární paměti tak, že rozdělili svůj program na úseky, které pak střídavě přenášeli pro zpracování do hlavní paměti počítače.
- Program byl rozdělen na části nazývané ***„overlaye“***, které byly programem (nazývaným ***„overlay manager“***) načítány do paměti a zpět na disk.
  - Každý ***overlay*** byl typicky modul, obsahující program i data.
- Součet všech „overlayů“ přítomných v paměti musel být menší než velikost fyzické paměti.
- To byl podstatný nedostatek a překážka pro každého programátora.

Činnost virtuální paměti



m je *vždy* menší než n

V opačném případě, k čemu VM?

„Cena“ virtuální paměti

Nedostupnost stránky se nazývá výpadek (miss).

- **Výpadek stránky:** Načtení dat z *disku* ☹ **(*latence* - milisekundy)**
- **Redukce „ceny“ VM :**
  - » Velikost stránek, aby se amortizovala dlouhá doba přístupu na disk
  - » Plně asociativní mechanizmus umísťování stránek, aby se snížila *četnost výpadků*.
- Obsluha výpadku stránek v software, protože mezi přístupy na disk je dost času, v porovnání s cache *(která je mnohonásobně rychlejší)*
- Použít *chytré softwarové algoritmy* pro umísťování stránek (je čas!)
  - » **Náhodné** umístění stránek **(Random)** se nepoužívá
  - » **Least Recently Used** je mnohem obvyklejší varianta

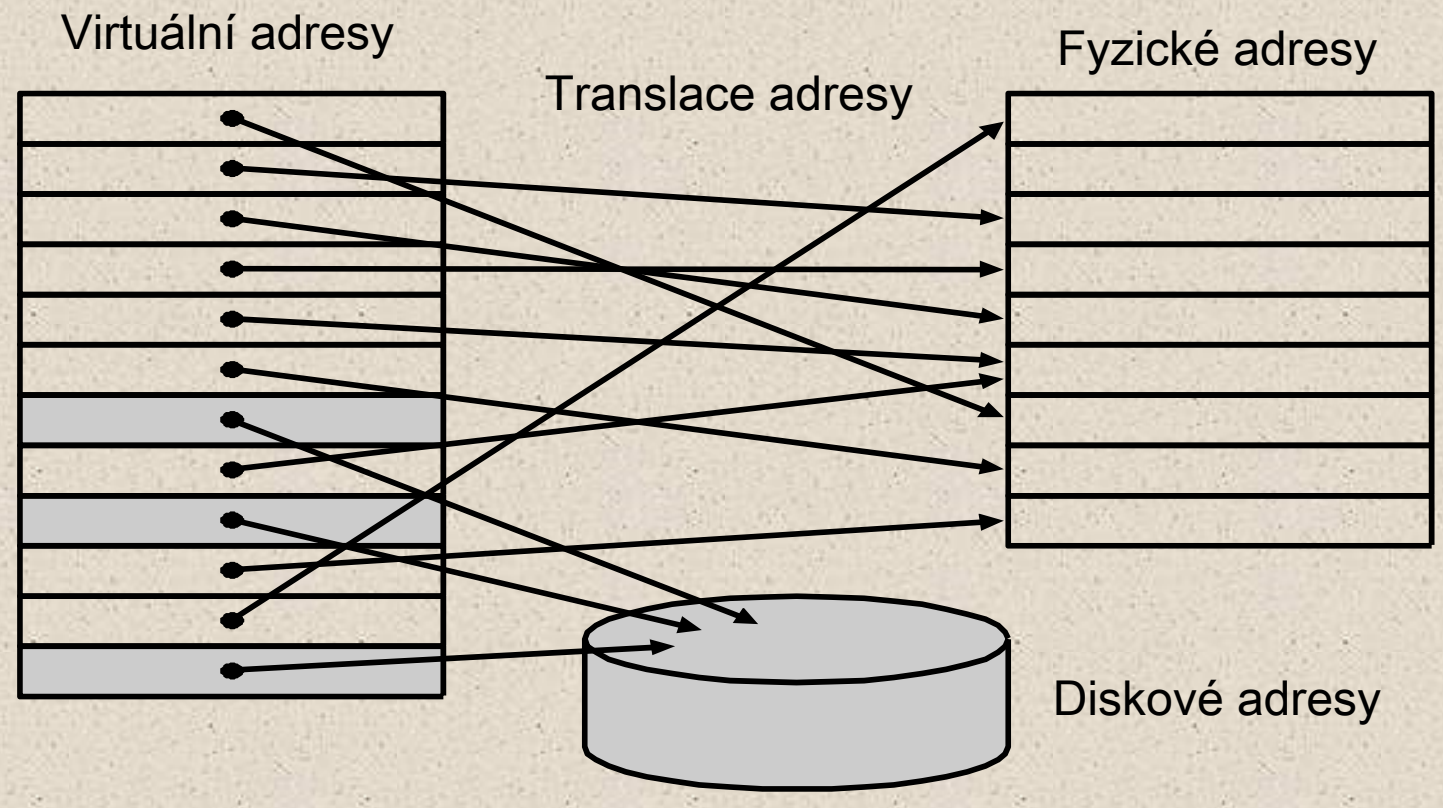
Terminologie

- Virtuální adresní (logický) prostor se dělí na *stránky*.
- VM miss se nazývá *výpadek stránky*.
- CPU vytváří *virtuální adresu* která je transformována kombinací hardware a software na *fyzickou adresu*, která je pak použita k přístupu do hlavní paměti.
- Tento proces se nazývá *dynamická transformace adresy* nebo *translace adresy*.
- Současné fyzické hlavní paměti jsou typicky dynamické paměti, jako sekundární paměť se používají magnetické paměti nebo flash technologie.

Relokace

- VM také zjednodušuje načítání programů s *relokací*. Mapuje virtuální adresy užívané programem na jiné fyzické adresy – to dovoluje umístit program do libovolného místa v hlavní paměti.
- Všechny současné systémy VM také relokují program jako soubor stránek. Pro načtení programu se pak nepotřebuje souvislý prostor v hlavní paměti; potřebujeme pouze dostatečný počet stránek ve fyzické paměti a vlivem systému VM jsou tyto stránky spojeny do souvislého prostoru (v logickém prostoru), i když tomu tak ve fyzické paměti není.

Relokace

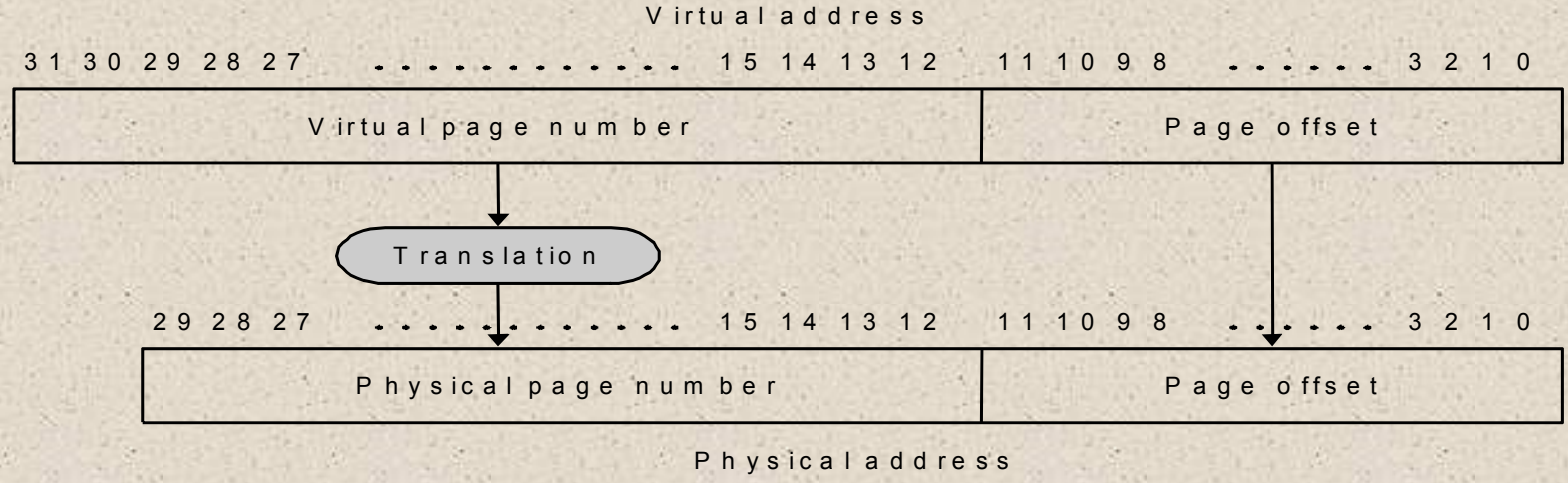


- Tento program obsazuje spojitý prostor ve virtuálním adresním prostoru, nikoliv však ve fyzickém.

Translace adresy

- Jak se provádí translace adresy z virtuálního adresního prostoru do fyzického?
- Každá adresa se rozdělí na pole „virtuální číslo stránky“ a pole „offset“ (adresa na stránce).
  - Virtuální číslo stránky se transformuje na fyzické číslo stránky.
  - Offset zůstává bez změny.
- Počet stránek virtuálního prostoru není obvykle roven počtu stránek fyzické paměti, bývá mnohem větší.
  - Velký počet virtuálních stránek (v porovnání s fyzickými) vytváří iluzi téměř neomezeného virtuálního prostoru.

Příklad translace adresy



- Velikost stránky je 2^{12} = 4K.
- Počet fyzických stránek je 2^{18}.
- Počet virtuálních stránek je 2^{20}.
- Fyzická paměť může mít nejvýše kapacitu 1GB, zatímco virtuální adresní prostor je 4GB.

Úvahy při návrhu

- Většina virtuálních paměťových systémů je navržena tak, aby se omezily výpadky stránek (page fault) – každý výpadek znamená přístup na disk, jehož zpracování trvá miliony cyklů procesoru.
- To vede při návrhu na celou řadu klíčových rozhodnutí…
  - Velikost stránky by měla být volena tak, aby se redukovala dlouhá doba přístupu. Typická velikost stránek je dnes 32K a 64K. Toto rozhodnutí je motivováno prostorovou lokalitou.

Úvahy při návrhu

- Redukce četnosti výpadků stránek má prioritu číslo jedna. Podporuje ji volnost při umísťování stránek.
- Výpadky stránek jsou typicky ošetřovány v software - pro umísťování stránek mohou být použity sofistikované algoritmy, jejichž použití je na úrovni hardware obtížné (v porovnání s výpadkem bloku u cache je více času).
- Techniky Write-through pro VM nelze použít, protože zápis trvá příliš dlouho. Systémy VM používají techniku Write-back.

Segmentace

- Alternativou pro stránkování je *segmentace*.
- Virtuální adresa se člení na dvě části - číslo segmentu a offset (adresa uvnitř segmentu).
- Každé číslo segmentu odkazuje na segmentový registr, který obsahuje fyzickou adresy segmentu. K tomuto pointeru je přičten offset a tak získáme aktuální adresu do fyzické paměti.
- Segmenty jednoho systému mohou mít různou velikost.
- Segmentace rozděluje adresu do dvou logicky oddělených částí, zatímco stránkování dělá hranici mezi číslem stránky a offsetem neviditelnou pro programátory a pro kompilátor.

Nyní zpět na stránkování...

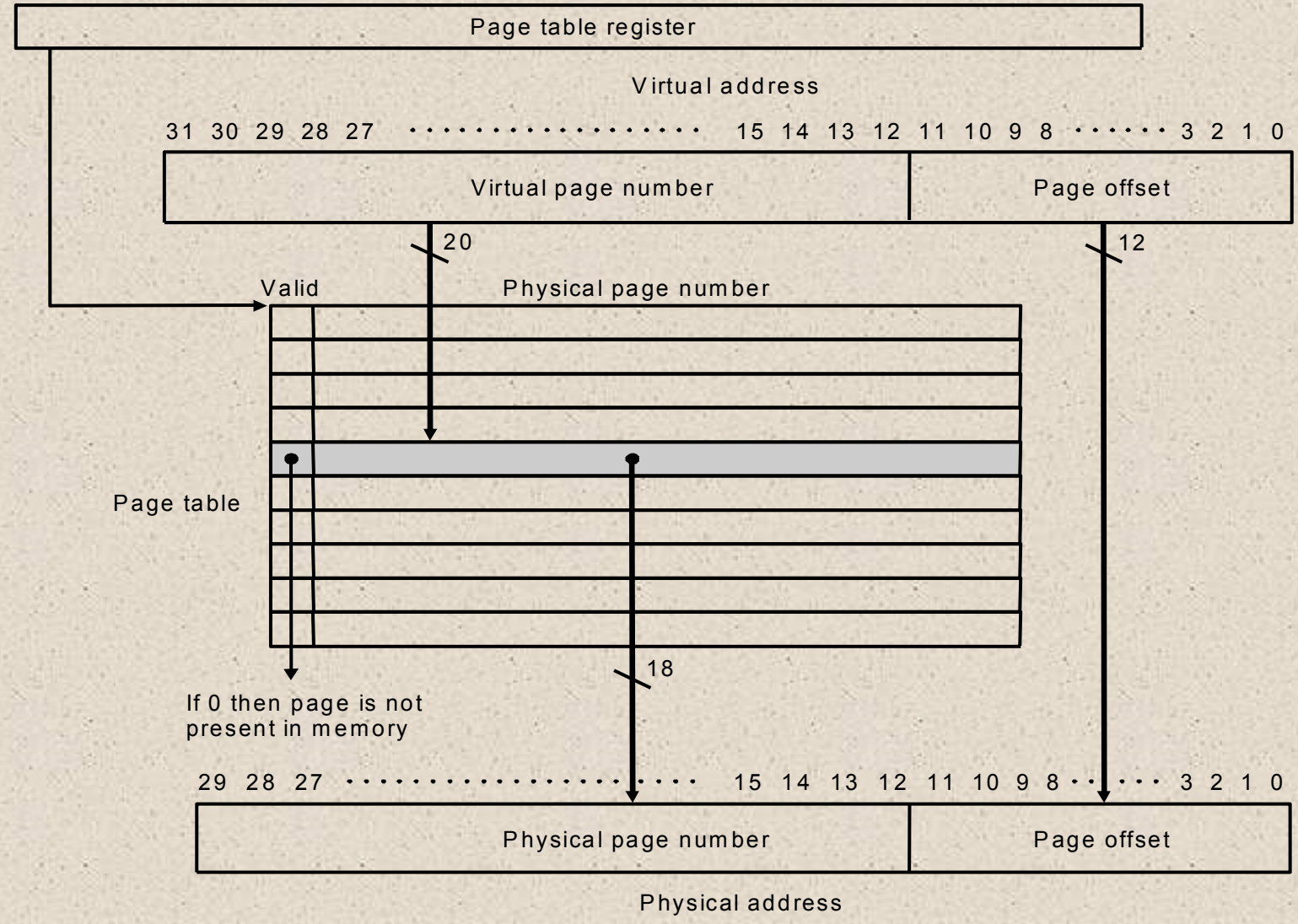
Umístění stránky

- Jak se umístí stránka do fyzické paměti?
- Umístění stránky je možné do libovolného místa, kam a kdy se stránka do hlavní paměti zapíše určuje operační systém.
  - To zajišťuje sofistikovaný algoritmus, součást OS, který sleduje využití stránky a naplánuje její přenos do vnější paměti, pokud se stránka delší dobu nepoužívá.
  - Volné umístění stránek poskytuje OS velkou flexibilitu při umísťování nové stránky.
  - Redukuje také četnost výpadků stránek.

Nalezení stránky ve fyzické paměti

- Jak je stránka nalezena, když se již nachází v hlavní paměti?
- Úplné prohledávání (analogie plně asociativní cache) není praktické.
- Ve virtuální paměti je na stránky přistupováno pomocí úplné tabulky, která indexuje paměť a nazývá se *tabulka stránek* (*page table*).
- Tabulka stránek je umístěna v paměti, vybírá se z ní pomocí čísla virtuální stránky a nalezená položka obsahuje index stránky do fyzické paměti.
  - Každý program může mít svoji vlastní tabulku pro svůj virtuální adresní prostor.

Tabulka stránek



Tabulka stránek

- Aby mechanizmus pracoval, musíme být schopni nalézt tabulku stránek. Proto je adresa počátku tabulky uložena do *registru stránkové tabulky* (*page table register*) programu. Sama tabulka leží v souvislé oblasti paměti.
- Stránková tabulka úplně mapuje virtuální adresní prostor a obsahuje mapování pro každou možnou virtuální stránku. Nepotřebujeme tagy.
- Bit platnosti má stejnou funkci jako v cache - je-li nulový, platná stránka není ve fyzické paměti a nastává výpadek stránky.

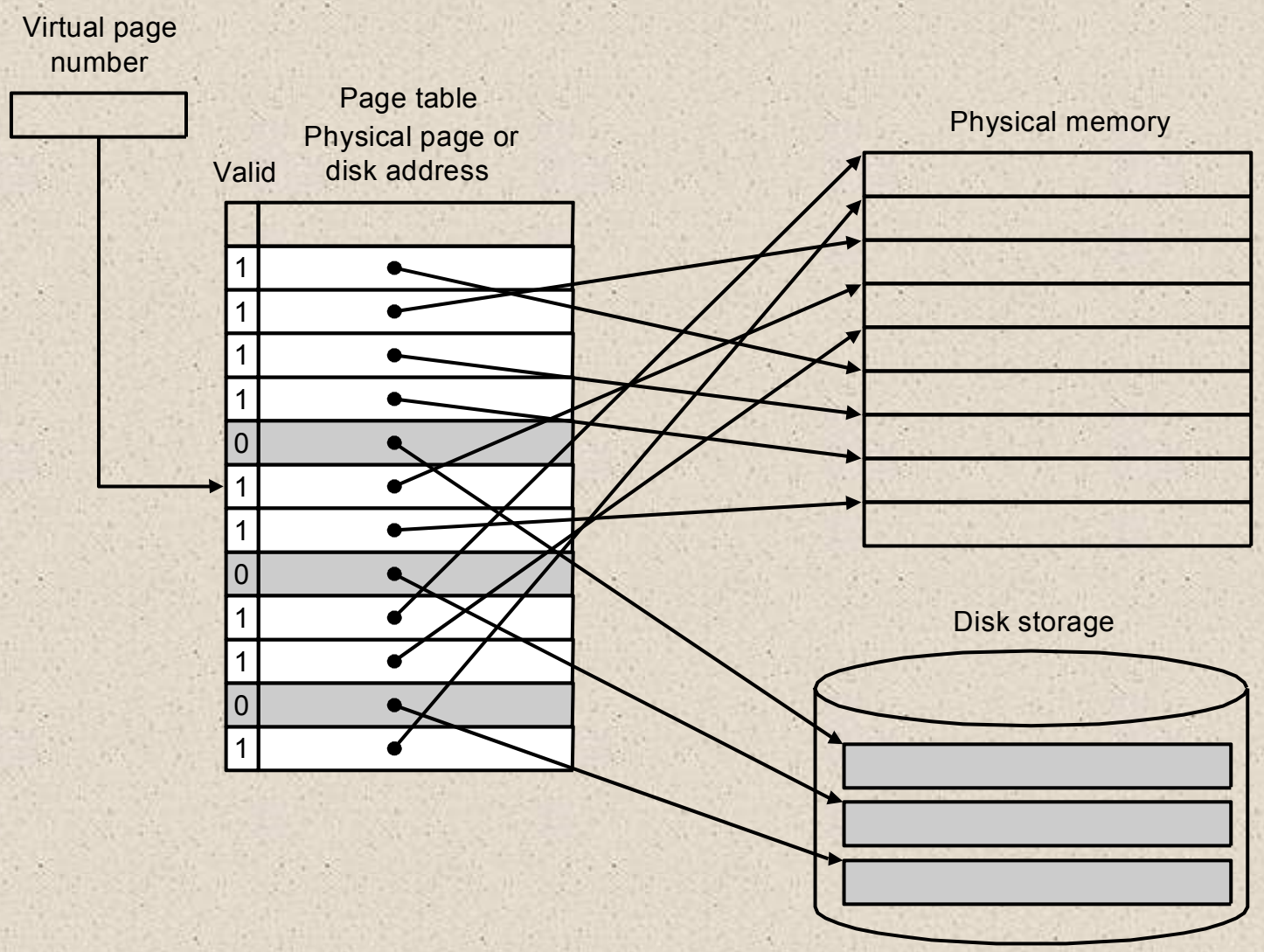
Ošetření výpadku stránky

- Co se stane při výpadku stránky?
- Operačnímu systému je předáno řízení, ten musí najít požadovanou stránku na disku a umístit ji do hlavní paměti.
  (rozdíl oproti cache !)
- Předání řízení a výměna se zajišťuje s využitím výjimek.
- Přirozeně je nutné udržovat informaci o poloze každé stránky na disku.
- Nevíme dopředu, kdy se bude stránka v paměti vyměňovat.
- OS vytváří datovou strukturu pro uložení každé virtuální stránky na disk.
  - Stránková tabulka obsahuje obvykle jak fyzickou adresu v paměti, tak adresu na disku.
  - Může to ale být i oddělená datová struktura, která existuje vedle tabulky stránek.

Výpadek stránky a disk

- Položka v tabulce stránek se nazývá *page descriptor*. Obsahuje celou řadu polí, která obsahují informace o dané stránce a využívají se pro organizaci výměny stránek.
  1. Aktivita stránky (jednobitový příznak)
  2. Pointer na počátek stránky ve fyzické paměti
  3. Poloha stránky ve vnější paměti
  4. Dirty bit (příznak modifikace) – (jednobitový příznak)
  5. Pole pro algoritmus výměny stránek (např. LRU)
  6. Pole ochrany proti nedovolenému přístupu
  7. …

Výpadek stránky a tabulka stránek



Implementace LRU

- O úspěšnosti VM rozhoduje také algoritmus výměny stránek. Jedním z nich je LRU.
- Předpokládejme, že systém VM používá algoritmus (LRU). OS vytvoří datovou strukturu, obsahující údaje, který proces užívá které fyzické stránky. Tato struktura také udržuje historii o přístupech na stránky, která je pak využita algoritmem LRU.
- Často se jedná jen o aproximaci (skutečný LRU by vyžadoval aktualizaci při každém přístupu).
  - Každá stránka používá referenční bit, který se nastaví po každém přístupu na stránky (R nebo W). OS periodicky nuluje tyto referenční bity – to dovoluje zjistit, na které stránky byl učiněn přístup během určitého časového intervalu.

Velikost stránkové tabulky

- Zabývejme se tím, jaké velikosti může tabulka nabýt.
- Předpokládejme, že máme 32-bitovou virtuální adresu, 4K stránky a 4 byty se vyžadují na jeden vstupní bod do tabulky stránek => můžeme spočítat celkovou velikost tabulky stránek…

$$PageTableEntries = \frac{2^{32}}{2^{12}} = 2^{20}$$

$$PageTableSize = 2^{20}\, PageTableEntries \times 2^{2}\, \frac{bytes}{PageTableEntries} = 4MB$$

- Potřebujeme tedy 4MB paměti pro každý aktivní program. Tento přístup se nejeví příliš realistický.

Omezení velikosti tabulky stránek

- Existuje celá řada metod, jak omezit velikost stránkové tabulky…
- Nejjednodušší technikou je použít registr, který je naplněn limitem velikosti tabulky pro každý proces.
  - Jestliže počet virtuálních stránek narůstá a překračuje limit, roste i tabulka stránek. To dovoluje přizpůsobit velikost tabulky požadavkům procesu.
  - Toto uspořádání vyžaduje, aby virtuální adresní prostor expandoval jenom jedním směrem.

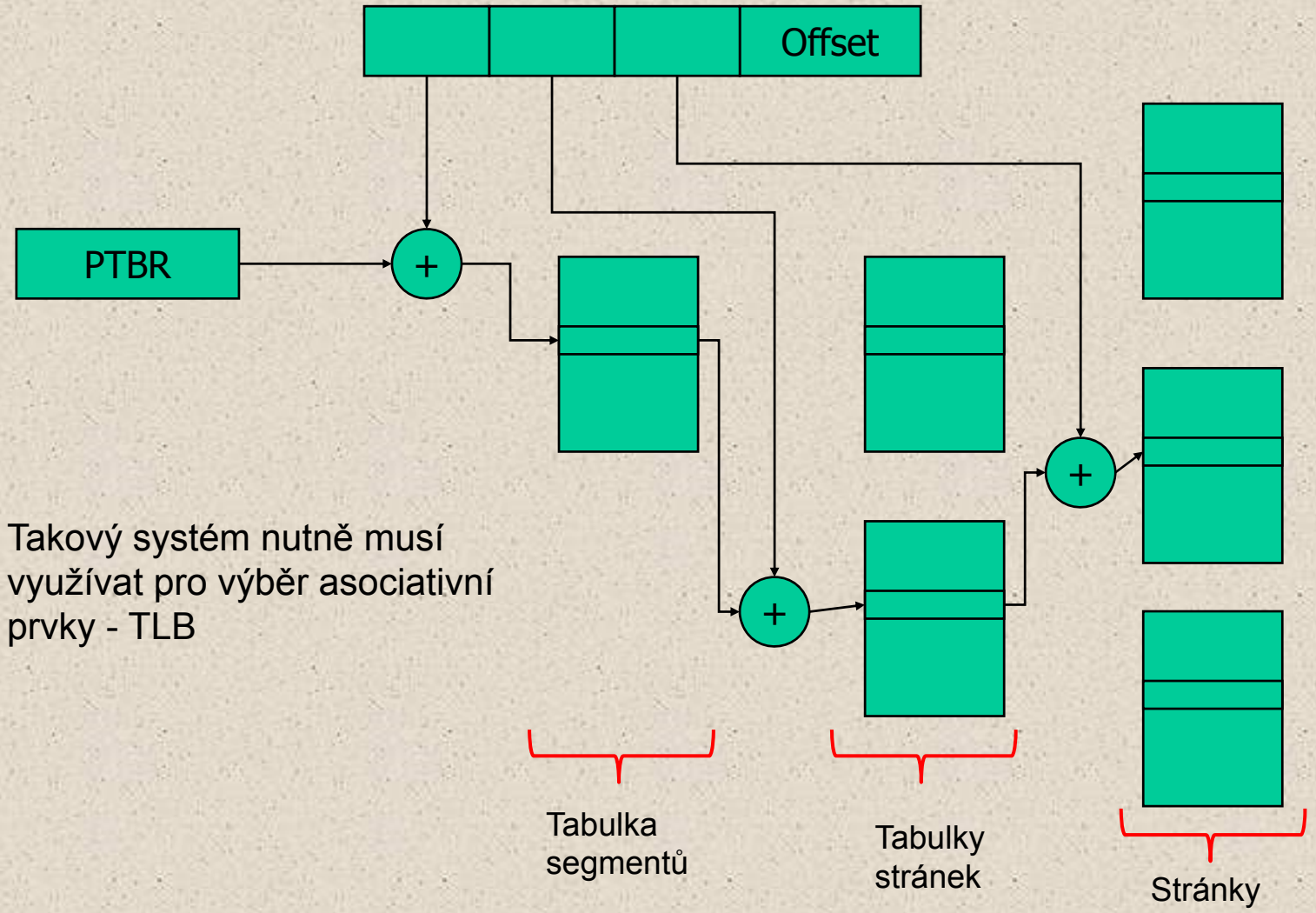
Složitější techniky

- Uvedená metoda není právě optimální. Potřeba nalezení efektivnějších přístupů.
  - Většina systémů obsahuje dvě oblasti, které musí expandovat, *zásobník* a *„halda“ (heap)*.
  - Některé systémy používají dvě oddělené rostoucí tabulky. Jednu, která se zvětšuje směrem nahoru a druhou, která se zvětšuje směrem dolů.
  - Využívají se dvě stránkové tabulky segmentů (jedná se o trochu jiný model segmentace, než o kterém již byla řeč). Nejvyšší bit adresy určuje která tabulka bude použita.
  - Toto uspořádáni pracuje velmi dobře ve většině případů. Neosvědčilo se v případě, kdy byly do virtuálního adresního prostoru činěny jednotlivé přístupy, namísto přístupů do kontinuální množiny adres…

Víceúrovňové stránkové tabulky

- Jinou metodu představují víceúrovňové stránkové tabulky.
  - První úroveň mapuje velké bloky fixní velikosti (také někdy nazývané segmenty). Tabulka první úrovně se také nazývá segmentová tabulka.
  - Segmentová tabulka udává, zda se nějaké stránky daného segmentu nacházejí v paměti. Když ano, obsahuje pointery na dané stránky segmentu.
  - Umožňuje to „řídké“ využití virtuálního adresního prostoru (násobné segmenty). Není třeba alokovat celou stránkovou tabulku. Systém se osvědčil u velmi velkých adresních prostorů s požadavky na nespojitou alokaci paměti.
  - Přístup do takové paměti je komplikovanější.

Víceúrovňový systém



Stránkování stránkových tabulek

- Většina moderních systémů dovoluje, aby byly *stránkovány* i *stránkové tabulky*.
- Využívá se stejného mechanizmu virtuální paměti. Stránkové tabulky jsou rovněž umístěny ve virtuálních adresních prostorech.
- Mohou ale vznikat nekončící rekurzivní posloupnosti výpadků stránek.
- Řešení jsou hardwarově-závislá a složitá. Proto jsou obvykle stránkové tabulky umístěny v adresním prostoru OS (kernel memory), v části paměti, která je stále v paměti přítomná a nepodléhá stránkování.

Zápis do paměti

- Systémy virtuální paměti používají strategii write-back pro zápis do paměti.
  - Rozdíl v době přístupu mezi cache a hlavní pamětí jsou desítky cyklů.
  - Rozdíl doby přístupu do hlavní paměti a na disk jsou miliony cyklů.
  - Strategie write-through, známá z úrovně cache, je nepoužitelná.

Zápisy do paměti a dirty bit

- Přináší další výhodu pro systém virtuální paměti.
  - Zápisy stránky na disk představují z hlediska cyklu procesoru velmi dlouhou dobu.
  - Vyplatí se uchovávat informaci, zda je nutné provést zpětný zápis stránky na disk v případě její výměny, nebo zda lze tuto diskovou operaci vynechat.
  - K tomu je určen *dirty bit*. Tento jednobitový příznak je nastaven v případě, že během přítomnosti stránky v hlavní paměti je na ni učiněn zápis. V případě výměny stránek, nebylo-li na stránku zapisováno, znamená to, že stránka v hlavní paměti je totožná s její kopií v hlavní pamětí. Výměna pak spočívá jen v načtení nové stránky.

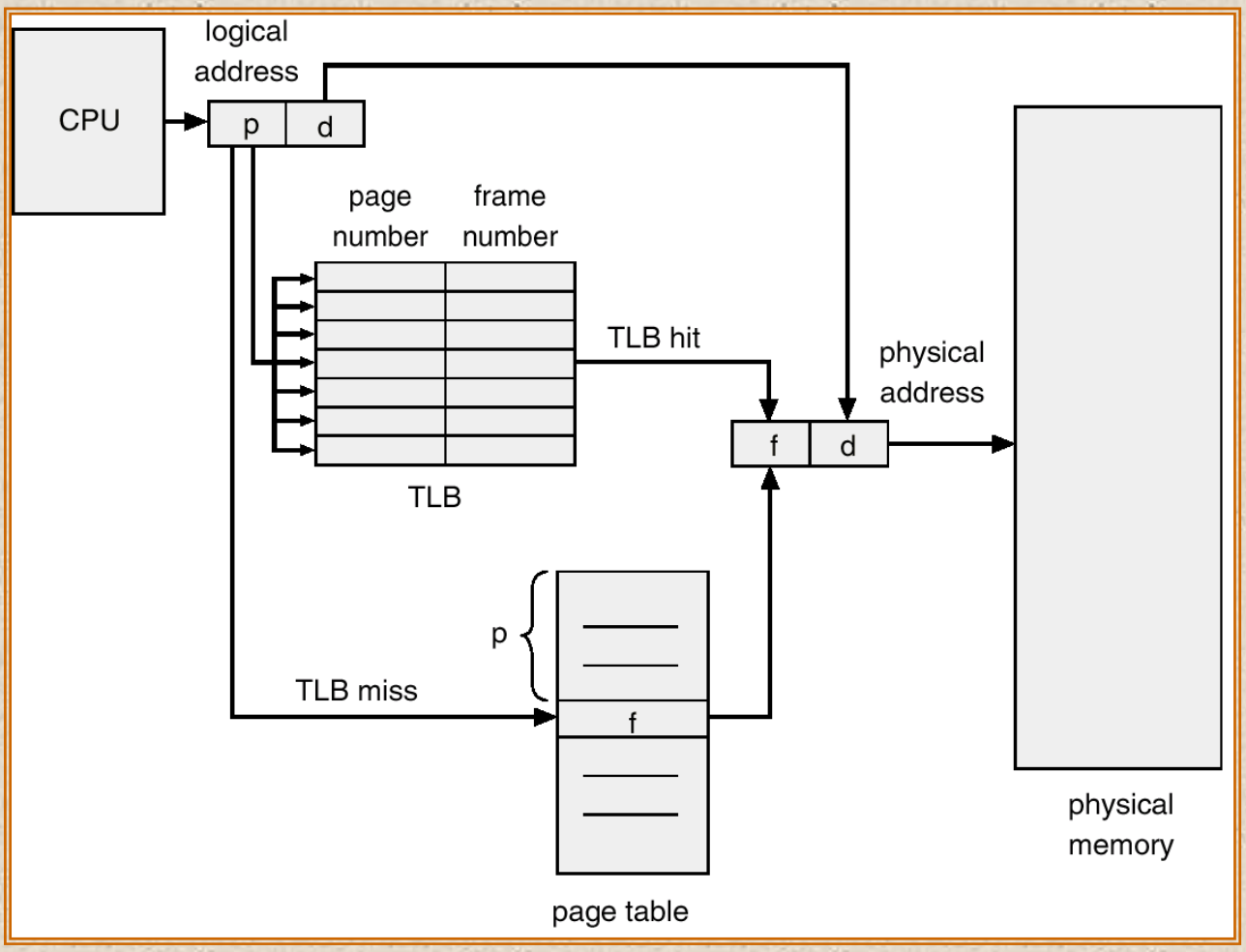
Rychlá transformace adresy

- Tabulky stránek jsou uloženy v hlavní paměti – to znamená, že přístup do paměti trvá dvojnásobnou dobu (jeden přístup pro získání fyzické adresy, druhý pro data).
- Klíčem pro zlepšení výkonu je *využití principu lokality* referencí do paměti – je-li jednou transformována adresa určité virtuální stránky, je velmi pravděpodobné, že výsledek transformace bude v zápětí opět třeba. Časová i prostorová lokalita.

Translation-Lookaside Buffer (TLB)

- Většina moderních systémů používá speciální cache, ve které jsou uloženy výsledky naposledy prováděných transformací adresy – nazývá se *Translation-Lookaside Buffer* (TLB).
- TLB je „cache“, která uchovává mapování stránek – každý vstup uchovává virtuální číslo stránky (tag) a číslo fyzické stránky (data uložená v této cache).
- Při přístupu do paměti se nejdříve provede přístup do TLB, nezačíná se přístupem do stránkové tabulky. Ta je použita jen v případě, že hledaná stránka není v TLB nalezena. Proto musí TLB obsahovat také různé informace, jako např. pointer na fyzickou stránku, dirty bit, atd.

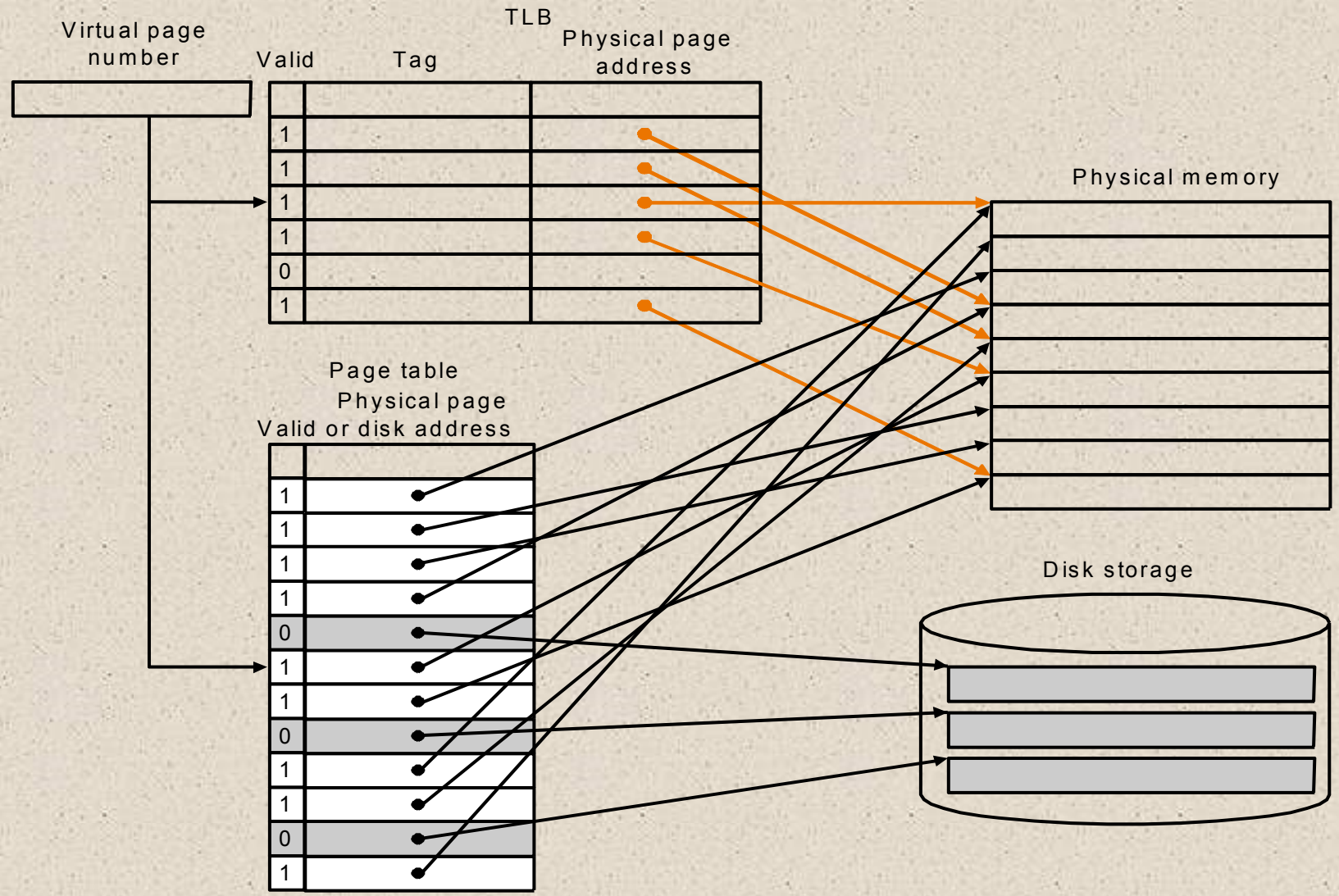
Virtuální paměť a TLB



Při neúspěchu v TLB pokračuje transformace výběrem ze stránkové tabulky

p … page
d …displacement (jinak offset)
f … frame

Virtuální paměť a TLB



Paměťové reference s TLB

- Při každé referenci se nejdříve hledá číslo virtuální stránky v TLB.
  - Při nalezení je zároveň získáno číslo fyzické stránky, nastaví se příznak reference a jedná-li se o zápis, nastaví se i dirty bit.
  - Při výpadku je nutno rozhodnout, jestli se jedná o výpadek stránky nebo jenom o výpadek TLB.
    - Jestliže je stránka přítomna v paměti, výpadek TLB indikuje, že translační informace v TLB chybí. V tomto případě CPU načte znovu translační informaci ze stránkové tabulky do TLB a výpočet pokračuje.
    - Není-li stránka v paměti, výpadek TLB znamená výpadek stránky. CPU výjimkou startuje obsluhu OS a stránka je načtena z disku do paměti.

Výpadek v TLB a výměna

- Výpadek TLB je mnohem častější než výpadek stránky, protože TLB má mnohem méně položek, než je stránek ve fyzické paměti.

- Potom, co nastane výpadek TLB a stránka je načtena, je třeba rozhodnout, která položka v TLB bude nahrazena. Návrháři využívají různé míry asociativity pro TLB.
  - Některé systémy mají TLB malé a plně asociativní, protože to zvyšuje úspěšnost. Protože TLB má malou kapacitu, není cena za plnou asociativitu tak vysoká.

Výpadky TLB a výpadky stránek

- Je jednoduché určit, který z uvedených dvou případů nastal.
- Zpracováváme-li TLB miss, nahlédneme do tabulky stránek, abychom vyšetřili vstupní bod stránky do TLB.
  - Má-li nalezený vstupní bod nastaven bit platnosti, můžeme získat fyzické číslo stránky z tabulky stránek a zapsat je do TLB.
  - Nemá-li nalezený vstupní bod nastaven bit platnosti, stránka se nenachází v paměti a jedná se o výpadek stránky.
- Nastane-li TLB miss, ale nikoliv výpadek stránky, lze toto situaci ošetřit v software nebo v hardware. Jde-li o výpadek stránky, je volán OS.

Ošetření výpadků v TLB

- Byl uveden obecný případ transformace virtuální adresy na fyzickou.
- Proces je velmi jednoduchý, pokud nastane TLB hit. Nastane-li miss v TLB, je situace složitější a zaslouží podrobnější rozbor.
- Zopakujme, že miss v TLB může indikovat dvě různé možnosti:
  1. Stránka je přítomna v paměti a pouze je třeba vytvořit chybějící přístupový bod v TLB pomocí tabulky stránek.
  2. Stránka není přítomna v paměti a pak je třeba předat řízení OS, který se musí vypořádat s výpadkem stránky.

Výpadky TLB a výměna

- Některé systémy mají velké TLB s malou nebo dokonce nulovou mírou asociativity.
- Není jednoduché vybrat optimální variantu.
  - U plně asociativní TLB je použití hardwarového LRU rozsáhlé a nepraktické.
  - Nelze použít ani složitější algoritmus v rámci OS, protože výpadky TLB jsou mnohem častější než výpadky stránek.
- Proto mnoho systémů provádí náhodný výběr položky TLB, určené pro výměnu (quick and dirty).

O výměně v TLB …

- Položky v TLB se nemění (se dvěma výjimkami). Výměna položky není náročná, mnoho se nestane.
  - Protože každá položka TLB má dirty bit a reference bit, pouze tyto dva bity je třeba kopírovat zpět do stránkové tabulky v okamžiku, kdy se položka z TLB odstraňuje.
  - Nic jiného se z TLB zachraňovat nemusí. Strategie write-back je efektivní, protože četnost výpadků TLB je celkem nízká.

Reálné systémy virtuální paměti

- Budeme se zabývat reálnou TLB a virtuální pamětí MIPS R2000/DECStation 3100.
  - Systém virtuální paměti používá stránky o velikosti 4K.
  - Adresní prostor je 32-bitů (číslo virtuální stránky je široké 20 bitů).
  - Fyzická adresa má stejnou velikost.
  - TLB má 64 položek, je plně asociativní a je sdílená pro data i instrukce. Každá položka zabírá 64 bitů a obsahuje 20-bitový tag, korespondující fyzické číslo stránky, bit platnosti, dirty bit a dvojici dalších administračních bitů.

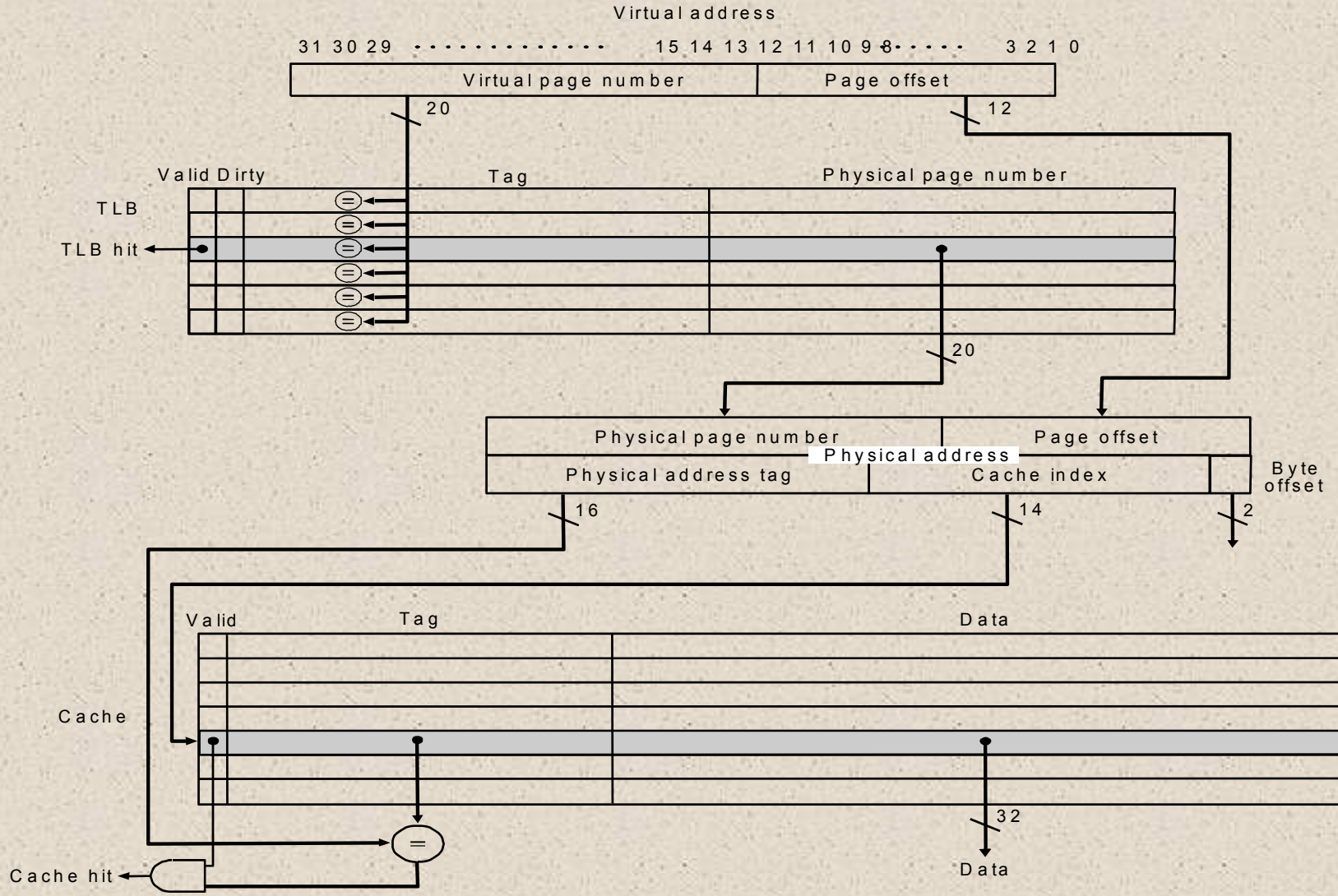
Výjimka výpadku stránky

- Nastane-li výpadek stránky, předá mechanizmus ošetření výjimky řízení OS (handler ošetření výpadku stránky).
- Výpadek stránky je rozpoznán přibližně v době cyklu, kterým se provádí přístup do paměti.
- Jakmile vstoupíme do handleru ošetření výjimky výpadku stránky (určeno registrem Cause), OS uloží celý stav probíhajícího procesu.
- OS zveřejní virtuální adresu, kde byl způsoben výpadek (využitím EPC nebo instrukcí a instrukci na EPC).

Výjimka - výpadek TLB

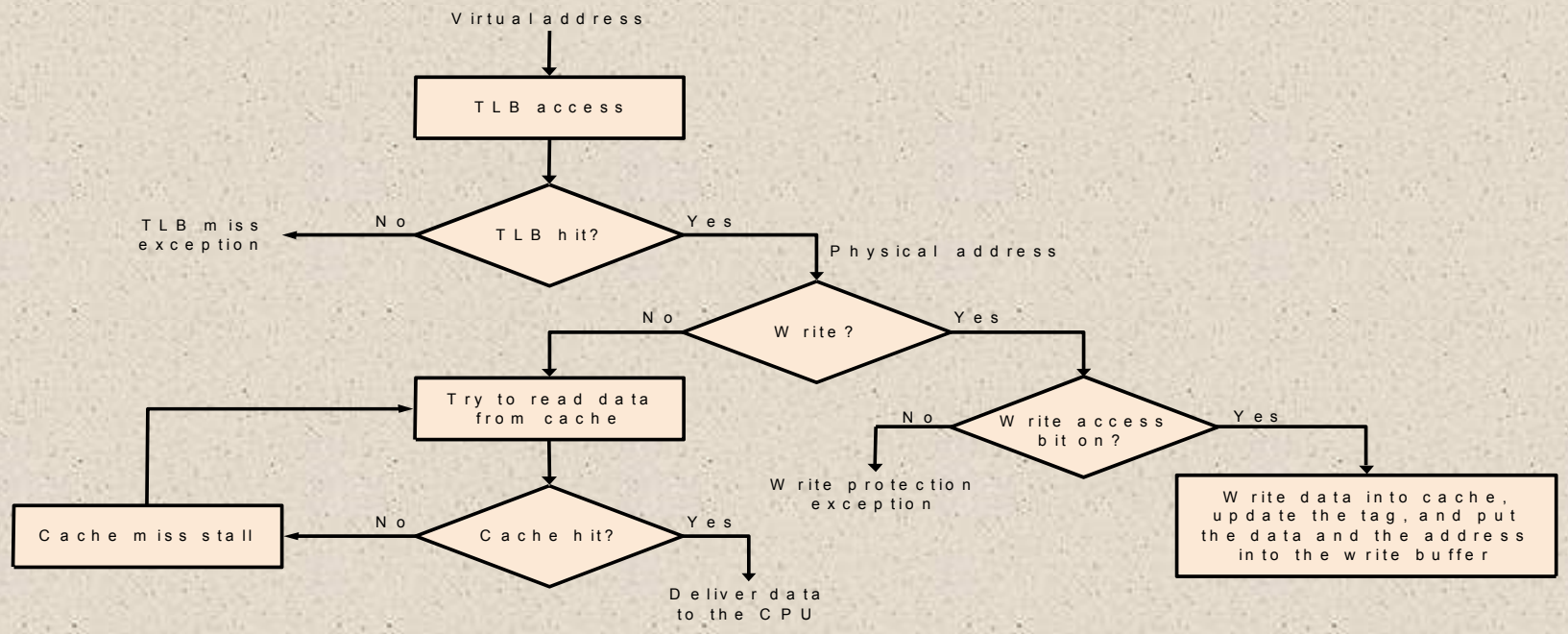
- Nastane-li výjimka výpadku TLB, hardware uloží číslo stránky reference do speciálního registru a generuje výjimku.
- OS zahájí obsluhu výpadku. Handler výpadku TLB zjistí ve stránkové tabulce uložené virtuální číslo stránky a registr stránkové tabulky.
  - Pravý výpadek stránky nastane, neobsahuje-li stránková tabulka fyzickou adresu.
- Užitím speciálního souboru instrukcí MIPS, OS aktualizuje TLB zapsáním fyzické adresy ze stránkové tabulky do nové položky TLB.

DECStation 3100 TLB a virtuální paměť



Přístup do paměti u DECStation 3100

- Hardware spravuje také index, který indikuje doporučenou položku TLB k výměně - index je vybrán náhodně.
- Obrázek znázorňuje celou proceduru přístupu do paměti pro DECStation 3100…



Interakce v paměťové hierarchii

- Jak je vidět, nejlepší případ nastává, jestliže je virtuální adresa transformována pomocí TLB a zaslána do cache, kde jsou požadovaná data nalezena.
- V nejhorším případě nastane všude výpadek, v TLB, v tabulce stránek i v cache.
- V takovém systému jsou všechny adresy transformovány na fyzické ještě před přístupem do cache.
- Cache je *indexována* a bloky označeny položkou *tag* podle *fyzické adresy*. To znamená, že jak *tag* tak *index* je odvozen podle *fyzické adresy* a nikoliv podle *virtuální*.

Interakce v paměťové hierarchii

- Alternativním řešením je indexovat cache pomocí virtuální adresy (*virtuálně indexována* a bloky označeny *virtuálním „tagem“*).
- V tomto uspořádání neprobíhá transformace adresy při běžných přístupech do cache, protože k adresování cache se používá virtuální adresa..
- Nastane-li výpadek cache paměti (cache miss), pak je virtuální adresa transformována na fyzickou, takže pak může být načten blok z hlavní paměti do cache..
- Výše uvedený návrh je složitější.

Interakce v paměťové hierarchii

- Podíváme-li se na vztah tří hardwarových jednotek a jejich interakci během přístupu do paměti, můžeme zjistit možné kombinace situací „hit“ a „miss“…

Cache	TLB	VM	Možné? Jak?
miss	hit	hit	možné – tabulka stránek se ale netestuje, je-li TLB hit
hit	miss	hit	TLB miss, položka ve stránkové tabulce nalezena, cache dává hit
miss	miss	hit	TLB miss, položka ve stránkové tabulce nalezena, cache dává miss
miss	miss	miss	TLB vykazuje miss, výpadek stránky, cache dává také miss
miss	hit	miss	nemůže nastat – v TLB nemůže nastat hit, pokud stránka není v paměti
hit	hit	miss	nemůže nastat – v TLB nemůže nastat hit, pokud stránka není v paměti
hit	miss	miss	nemůže nastat – data nemohou být v cache není-li stránka v paměti

Restart versus přerušení instrukcí

- MIPS instrukce jsou *restartovatelné* – jestliže některá způsobí výpadek stránky, je zastavena uprostřed provádění a OS ošetří výpadek stránky.
- Jakmile je výpadek stránky ošetřen, instrukce je startována od počátku.
- Toto lze provést u procesorů s jednoduchým instrukčním souborem jako je MIPS. U složitějších architektur musí být instrukce přerušena, uložen kontext a později se dokončuje rozpracovaná instrukce.
  - Toto je nutné, protože některé instrukce mohou pracovat s tisíci datovými slovy (např. blokové přenosy).
  - Vyžaduje to pak mnoho HW prostředků pro uložení stavu a složité interakce mezi HW a OS.

Ochrana paměti

- Jednou z velmi důležitých vlastností virtuální paměti je možnost sdílení jednoho paměťového prostoru více procesy. Vyžaduje to ale ochranu paměti mezi jednotlivými procesy a OS.
  - Žádný proces nesmí zapisovat do adresního prostoru jiného procesu.
  - Žádný proces nesmí číst z adresního prostoru jiného procesu.
  - Procesy musí mít chráněný svůj virtuální adresní prostor. Jinak, VM má své výhody i nevýhody.
- VM je klíčem k *izolaci procesů*.
- Pole bitů pro ochranu jsou součástí *page descriptorů*, popř. *segment descriptorů*

Implementace izolace procesů

- Každý proces má svůj virtuální adresní prostor.
- Jestliže OS organizuje stránkové tabulky tak, že nezávislé virtuální stránky mapuje do disjunktních fyzických stránek (nesdílí se), pak žádný proces nemůže přistupovat k datům jiného procesu.
- Žádný proces nesmí manipulovat se svými stránkovými tabulkami, ale OS může.
- Z tohoto důvodu umísťují moderní operační systémy stránkové tabulky do vlastního adresního prostoru (kernel memory).

Závěr

- Použití cache zlepšuje efektivitu paměti:
  - Omezuje často se vyskytující operace read/write na přístup do rychlé paměti
  - „Shlukuje“ přístupy registr-hlavní paměť
  - Mění charakter chování procesoru vzhledem ke sběrnici
  - Vytěžuje I/O pipeline a tak *maximalizuje propustnost hierarchického paměťového systému*
- Použití virtuální paměti je výhodné, protože:
  - Mapuje velký symbolický adresní prostor na malý fyzický adresní prostor – mechanizmus podobný cache

Příště: I/O a sběrnice