Hardware pro ochranu virtuální paměti a procesů

Břetislav Bakala

Mechanismy ochrany paměti

- Více programů běží současně na jednom stroji a mají **současné nároky** na jeho využití – sdílení systémových prostředků, což vede k požadavkům na jejich **ochranu**
- Virtuální paměť založená na **segmentaci a stránkování** včetně TLB (obsahující tabulků přístupů do cache) je **primární mechanismus** chránící jednotlivé procesy mezi sebou.

Tyto mechanismy byly probrány detailněji v kurzu Architektura počítačů I.

Přepínání procesů

- Proces potřebuje ke svému běhu **„životní prostor“** – systémové prostředky
- Tyto instance je **nutné sdílet** s ostatními procesy a je potřeba zajistit přepínání z jednoho procesu na jiný
- Přepínání procesů – **řízení přepnutí** mezi procesy
- Přepínání kontextů – **přepínání stavů** (množiny informací nutných k **obnovení činnosti** procesu poté, co byl přerušen)
- Architektura procesoru ve spolupráci s operačním systémem umožňují procesům **sdílet hardware aby procesy nebyly vzájemně ovlivněny**

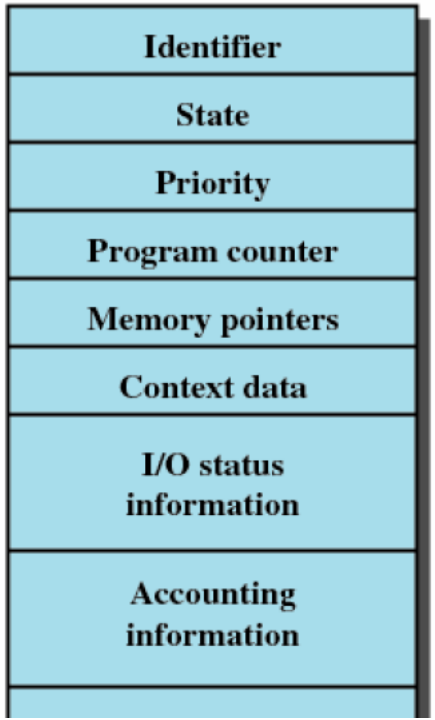
Popis procesů - PCB

PCB =(Process Controll Block) - datová struktura spravovaná jádrem OS

- **PID**
- **Identifikace stavu**
- **Priorita**
- **Obsah registrů** procesoru (Program Counter, Memory Pointers - ukazatele na kód a data procesu a dále paměťové bloky sdílené s ostatními procesy, Context Data = data v registrech procesoru)
- **Informace o I/O** zařízeních (I/O requests, I/O zařízeních, souborech)

přidělených procesu

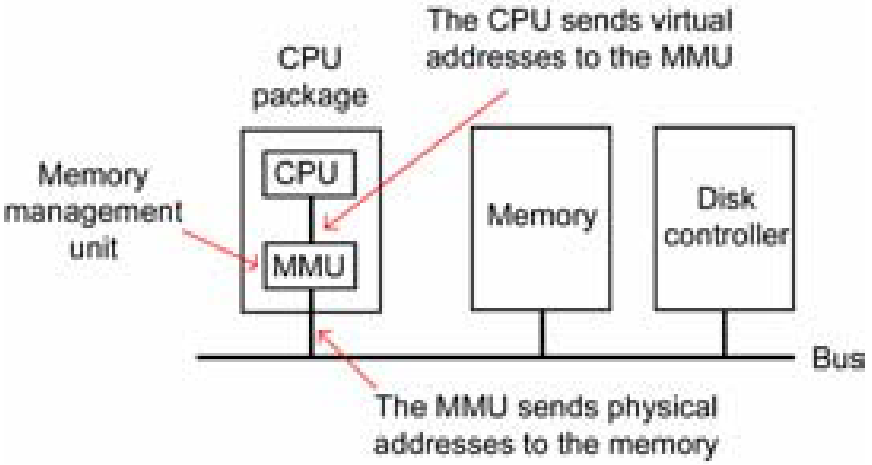
- **Accounting** („účtovací informace“, čas spuštění, kolik procesorového času spotřeboval, aj.



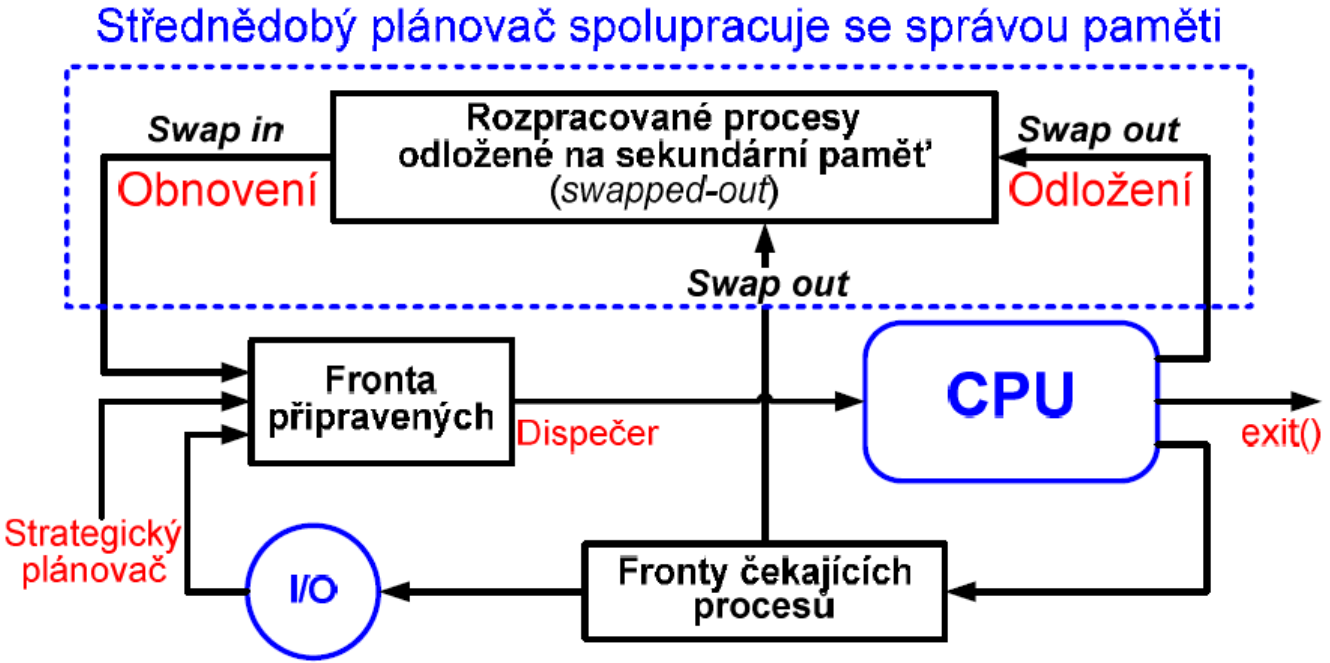
Memory Management Unit (MMU)

- Překládá virtuální adresu na fyzickou.
- Řeší výpadek stránky (Page fault)

– OS nejdříve definuje, který rámec fyzické paměti je třeba uvolnit, a pak do něj nahraje obsah požadované virtuální stránky z disku.



Odkládání procesů do Swap



HW zajištění oprávnění přepnutí

HW musí obsahovat následující minimum:

- Umožnit běh procesů v **módu jádra** OS (proces supervisoru) nebo v **módu uživatele**
- Pro proces v roli uživatele umožnit **natavení pouze číst**. Tento stav určuje user/supervisor stavový bit, bit možnosti **spuštění procesu výjimky**, a **informace o ochraně** dané paměti. Uživatel tak nemůže přímo ovlivňovat řízení procesů v operačním systému, vypnout výjimky nebo změnit ochranu paměti.
- **Mechanismus přechodu** z uživatelského režimu do režimu supervizora a naopak

HW zajištění oprávnění přepnutí

HW musí obsahovat následující minimum:

- Umožnit běh procesů v **módu jádra** OS (proces supervisoru) nebo v **módu uživatele**
- Pro proces v roli uživatele umožnit **natavení pouze číst**. Tento stav určuje user/supervisor stavový bit, bit možnosti **spuštění procesu výjimky**, a **informace o ochraně** dané paměti. Uživatel tak nemůže přímo ovlivňovat řízení procesů v operačním systému, vypnout výjimky nebo změnit ochranu paměti.
- **Mechanismus přechodu** z uživatelského režimu do režimu supervizora a naopak

Mechanismus přepnutí rolí

- Přechod z uživatelského režimu do režimu supervizora – **systémové volání** - zvláštní instrukce, přenáší **kontrolu do vyhrazeného místa** v prostoru kódu **supervisora**. **Stav** počítače je **uložen v bodě systémového volání** a procesor se **přepne do módu supervizora**. Návrat do uživatelského režimu je jako návrat podprogramu, který obnoví původní role.

- Jedná se o **omezení přístupu** k paměti pro ochranu stavu paměti procesu, **bez swapování procesu** na disk, ke ktrému by došlo při přepínání kontextu.

Ochrana stránkované paměti

Přidání ochrany každé stránce ve virtuální paměti, ochrana je přidána do každé tabulky stránek.

Ochrana stránkování určuje, který uživatelský proces může:

- **Číst danou stránku**
- **Zapisovat na danou stránku**
- **Může provádět kód na dané stránce**

Proces **nemůže číst ani zapisovat** na stránce, která **není v tabulce stráne**k (tabulku spravuje pouze operační systém).

Využití funkce TLB

- Při **stránkování** virtuální paměti každý přístup do paměti trvá nejméně dvakrát déle (**získání fyzické adresy, získání dat**) – neekonomické

Využití principu lokality:

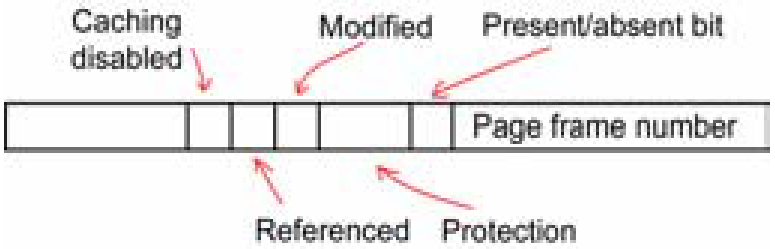
- pokud **přístupy** do paměti mají určitou **lokalitu**, musí ji mít také **adresy překladů přístupů.**
- **Udržováním** těchto překladů je nutný **přístup k paměti pouze zřídka**
- **Mezipaměť překladů** adres je označována jako **TLB**

Záznam TLB

Záznam v TLB je podobný záznamu v cache, kde **tag** obsahuje **část virtuální adresy** a **datová část** obsahuje **fyzickou adresu stránky**, pole **ochrany**, bit **platnosti** stránky

Valid	Virtual page	Modified	Protection	Page frame
1	140	0	RWX	31
1	12	0	R	12
1	25	1	R X	13
1	256	0	RWX	23
1	2	1	RWX	5
1	311	1	RWX	78

Struktura v tabulce stránek



- Page frame number
- Present/absent bit
- Protection bits - reading, writing, executing.
- Modified - HW při zápisu nastaví dirty bit na 1.

Když OS uvolňuje rámec stránky:obsah stránky uložit na disk Modified = 1. jinak nahrát novou stránku.

- Referenced bit – přístup ke stránce bit = 1. použíto pro algoritmy náhrady stránek.
- Caching disabled bit - stránky mapovány na registry periferních zařízení. Pokud čekáme na V/V (např. v cyklu), musíme použít hodnoty z fyzických HW registrů, nikoliv (starý) obsah v paměti.

Vliv ochran na architekturu

Příklad:

- Příkaz 80x86 POPF v uživatelském režimu umožňoval změnit všechny příznaky **kromě IE**, kdežto v režimu supervisoru byla změna IE povolena
- Při spuštění virtuálního stroje v uživatelském režimu došlo ke konfliktu – **VM očekával možnost změny IE**
- **Rozšíření architektury** 80x86 podporující virtualizaci tento problém vyloučily.

Koherence dat v cache

- **Zajištění stejnosti** dat z hlavní **paměti** v kopiích na **všech úrovních cache**

- Zvýšení nebezpečí nekoherence ovlivňuje:
- Již zmíněné relativně nízké nebezpečí změny dat při **přístupu do cache přes write buffer**
- Víceprocesorový systém - výkon multiprocesního programu závisí na výkonu **systému při sdílení dat**.
- **I/O zařízení** - je třeba se nekoherenci vyhnout

Koherence při I/O operacích

- Pokud by vstup zadával data do cache a výstup četl data z cache L1, **zařízení I / O a procesor by viděl stejné údaje** – byl by to zásah do procesoru, který by přitom musel čekat.
- Vhodnější je při **I/O operacích** pracovat přímo s **hlavní pamětí** (plní funkci **I/O vyrovnávací paměti**
- **Při zápisu** a použití **write-thru** cache by hlavní paměť měla aktuální hodnotu z cache. Ta se ale dnes používá **na úrovni L1**(proto se využívá **zápis řízený procesorem**), na **vyšších úrovních cache** se používá zápis **write back**
- **Při vstupu** musí **OS zajistit**, aby cache obsahovala **nekešovatelný blok** a mohl být vždy použit. (může také nejprve vyprázdnit cache)
- **HW řešení** zkontroluje I/O adresu s adresou v cache a **zajistí její uvolnění**

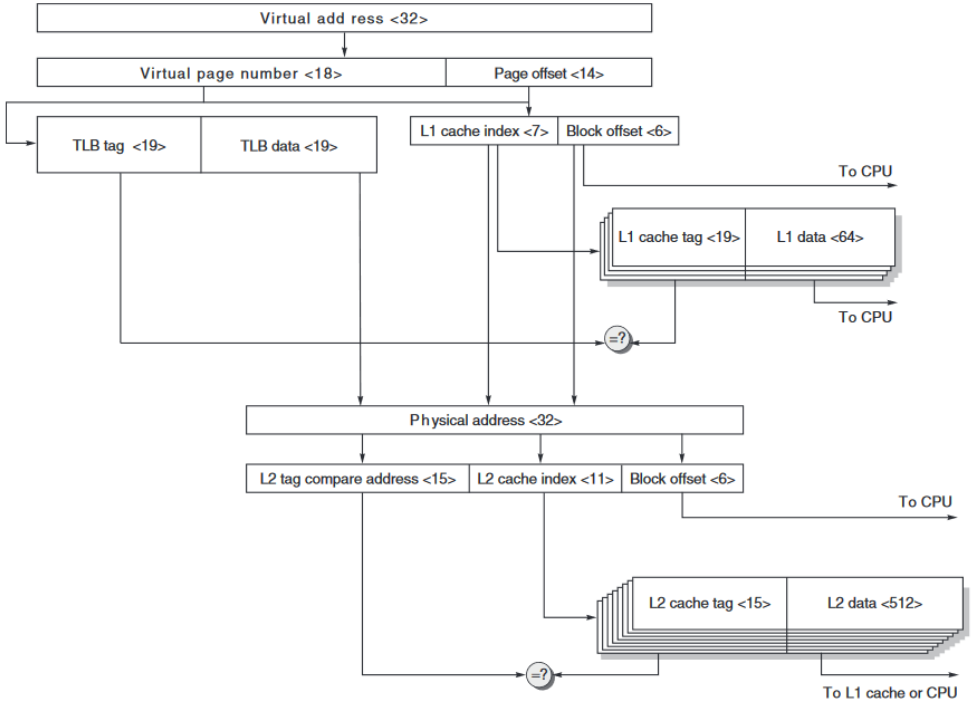
Příklad architektury ARM Cortex I.

- Instrukční sada ARMv7, jádro IP (Intellectual Property), pro embedded, PMD
- Může být vícejádrová, obsahuje aplikačně zaměřené procesory(video kodek)
- Může mít **speciální paměťové a I/O rozhraní** optimalizované např. pro Apple (Cortex-A8 IP)
- Procesory se stejným jádrem jsou proto realizovány **do různých čipů**
- Cortex-A8 poskytuje zpracování **dvou instrukcí v jednom hodinovém cyklu** s frekvencí až 1 GHz
- Čtyřcestná Cache L1 16 kB pro instrukce, 32 kB pro data, používá way prediction a náhodný přepis. **Přístup k paměti trvá 1 hodinový cyklus**. Je **virtuálně indexovaná a fyzicky tagovaná**. Velikost bloku 64 kB.

Příklad architektury ARM Cortex II.

- Cache L2 je volitelná, osmicestná, s velikostí 128 kB – 1MB, organizovaná do čtyř bank umožňující paralelní přenos, velikost bloku 64 kB. Je fyzicky indexovaná a tagovaná
- Oddělený TLB (I a D) plně asociativní se 32 záznamy a velikostmi stránek 4 kB, 16 kB, 64 kB, 1 MB, 16 MB, TLB miss řešený HW

Hiearchie paměti ARM Cortex



Příklad architektury Intel i7

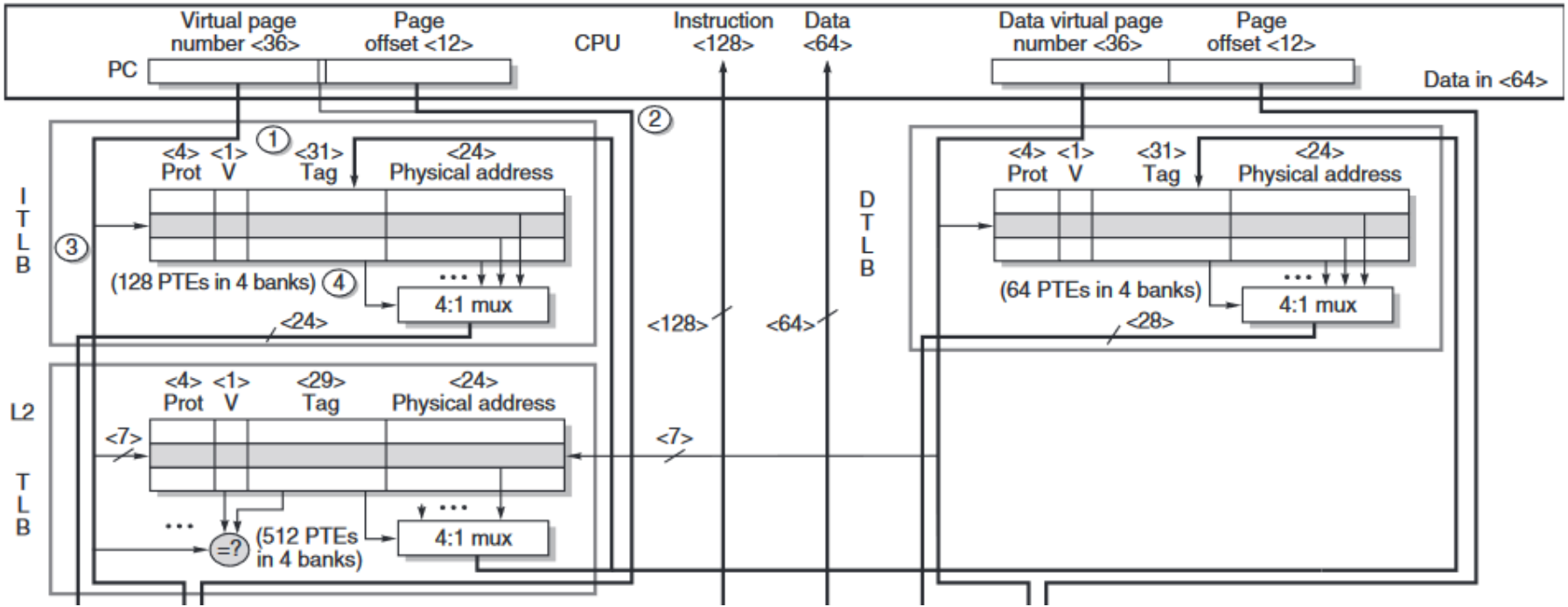
- 48-bit virtuální adresa, 36-bit fyzická adresa
- Cache L1 je virtuálně indexovaná a fyzicky tagovaná
- Cache L2, L3 je fyzicky indexovaná a tagovaná
- Všechny cache jsou typu write back, velikost bloku 64 B
  (LRU - Least Recently Used)

Characteristic	L1	L2	L3
Size	32 KB I/32 KB D	256 KB	2 MB per core
Associativity	4-way I/8-way D	8-way	16-way
Access latency	4 cycles, pipelined	10 cycles	35 cycles
Replacement scheme	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU but with an ordered selection algorihtm

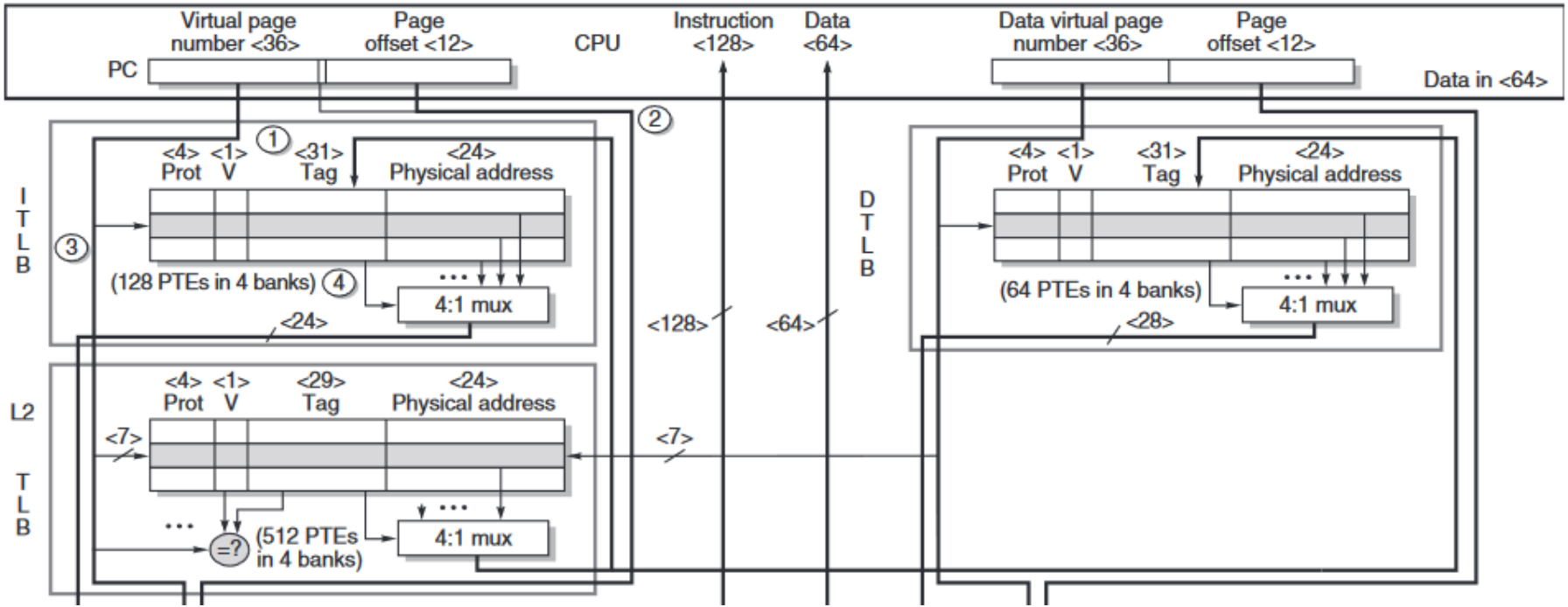
Struktura TLB - Intel i7

Characteristic	Instruction TLB	Data TLB	Second-level TLB
Size	128	64	512
Associativity	4-way	4-way	4-way
Replacement	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU
Access latency	1 cycle	1 cycle	6 cycles
Miss	7 cycles	7 cycles	Hundreds of cycles to access page table

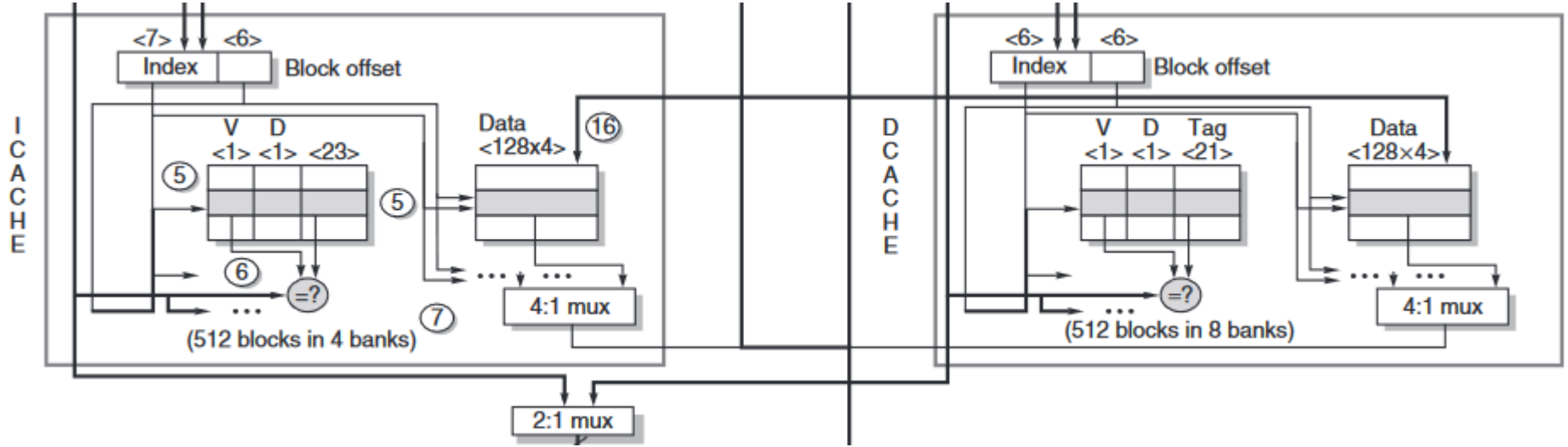
Paměťová hierarchie - Intel i7 I.



Paměťová hierarchie - Intel i7 I.



Paměťová hierarchie - Intel i7 II.



Paměťová hierarchie - Intel i7 III.