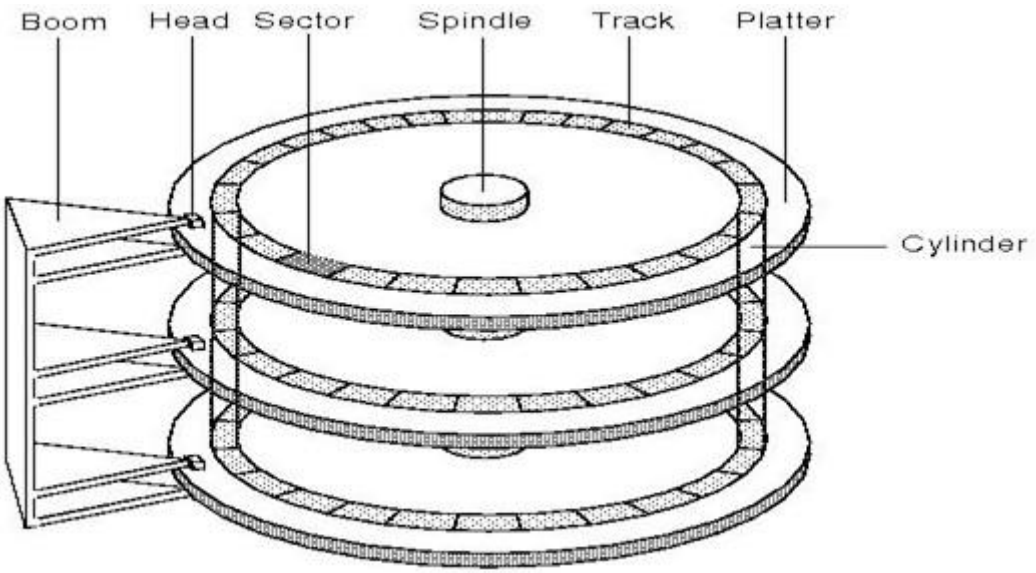
Správa paměti

Fyzická a lineární paměť

Na fyzické vrstvě je adresace paměti daná hardwarem. U rotačních disků jsou paměťové bloky definované cylindrem, hlavou a sektorem, viz obrázek



Obrázek 1: fyzická adresace rotačních disků. zdroj superuser.com

Adresu tvoří 24bitové číslo, kde první 10 b odpovídalo pořadí cylindru od středu, dalších 8 b pořadové číslo hlavy (plotny, na které se pohybovala) a 6 b na sektor: ovladače potřebovaly správné nastavení těchto údajů vč. velikosti sektoru (typicky 512 B).

Tato adresace je hardwarově závislá a např. u SSD nemá smysl. Proto moderní firmware (umístěný na řadiči paměťového zařízení) převádí adresu na **lineární** (LBA, *logical block addressing*). Okolní HW pak prostřednictvím tohoto firmware vidí pouze lineární adresu.

Pro rychlý přístup k paměti se využívají registry. Na mikroprocesorech jsou to zejména:

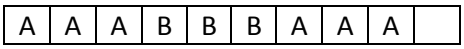
- CS (*Code Segment*) ukazuje na začátek segmentu paměti vyhrazeného pro kód (fyzicky často EEPROM)
- DS (*Data Segment*) ukazuje na segment paměti vyhrazený pro data programu
- ES (*Extra Segment*) ukazuje obvykle na paměť typu FLASH (nevolatilní, pomalá)
- SS (*Stack Segment*) ukazuje na místo v paměti pro zásobník

Tyto registry nastavuje OS a jsou pro program neměnné: obsahují též údaje o oprávnění k vykonávání instrukcí (některé jsou privilegované a nemůže je spustit obyčejný proces, např. načtení globální tabulky deskriptorů, viz níže).

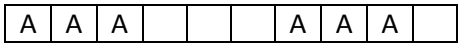
Virtuální paměť

Při alokaci dynamické paměti může dojít k tzv. setřásání, viz umělý příklad postupného alokování a uvolňování 10B volné paměti

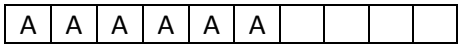
1. Proces A alokuje 3 B paměti, poté proces B alokuje další 3 B paměti, poté proces A alokuje další 3 B paměti



2. Proces B uvolní alokovanou paměť



3. Proces B potřebuje alokovat 4 B paměti, pro tu ale není souvislé místo k dispozici. OS ale může přesunout druhý blok A k prvnímu, čímž se místo uvolní



Druhý blok paměti A tedy změnil svou lineární adresu. Proto CPU poskytuje další abstrakci nad pamětí: **virtuální paměť**. Při vykonávání strojové instrukce se pracuje s virtuální pamětí, která se překládá na lineární pomocí MMU (*memory management unit*). Tato správa nepřesouvá jednotlivé byty, ale jejich bloky stejné velikosti (**stránky**), obvykle 4 kB velké.

Každý proces má tedy od OS přidělen adresář stránek (*page directory*, který je uchováván též v registru CR3) a segmentové registry, které ukazují do tabulky deskriptorů (GDT, global descriptor table pro OS; a LDT, local descriptor table pro proces). OS nastavuje a přepíná procesy změnami segmentových registrů a LDT. Záznam v této tabulce obsahuje počátek a velikost segmentu a příznaky a oprávnění pro práci s ním.

Pokud je tedy v programu 32bitová virtuální adresa (např. v ukazateli), překlad probíhá následovně:

1. Zjistí se počátek LDT z registru CR3
2. Horních 10 bitů je pozice v adresáři stránek, která obsahuje 32bitovou adresu, na které se nachází tabulka stránek pro daný proces
3. Dalších 10 bitů z této tabulky obsahuje na pozici v tabulce stránek, kde se nachází lineární adresa stránky
4. Posledních 12 bitů se nepřekládá, je to pozice v rámci stránky

TLB

Výšeuvedený postup má za následek, že při každém čtení z paměti musíme provést tři čtení (načtení adresáře stránek a tabulky stránek pro daný proces), což je o 2-3 řády pomalejší proces než zpracování instrukce. Proto MMU obsahuje TLB (*translation lookaside buffer*) což je malá vyrovnávací asociativní paměť pro naposledy překládané adresy. Záznam v této paměti se skládá z klíče (tagu) tvořený překládanou částí virtuální paměti a z fyzické paměti.

Překlad pak probíhá následovně:

1. MMU se podívá, jestli překládaný tag není v TLB. Pokud ano (cache hit), paměť se vůbec nepoužije
2. Pokud tam tag není (cache miss) vybere se nejdéle nepoužitý záznam (LRU, *least recently used*), který se odstraní. Lineární adresa z tabulky stránek se doplní na uvolněné místo v TLB

Virtuální paměť může být též větší než reálná paměť. OS v tom případě použije LRU na blok paměti nejdéle nepoužívaného procesu, kterou zapíše na disk (*swap*). Pokud tento proces ke své paměti zkusí přistoupit, řadič MMU vygeneruje HW vyjímku *Page Fault*, kterou pak zpracuje OS tím, že použitím LRU provede swap s jiným procesem.

Tato operace je však o 5-6 řádů pomalejší než vykonání instrukce (procesor stihne mezitím vykonat miliony instrukcí), proto OS obvykle mezitím přepne na jiný proces. Pokud ale tento proces chce také přistupovat k disku, dojde k citelnému zpomalení počítače.

Správa paměti

Správa paměti je v režii OS a dělí se na

- **automatická**: proměnná je alokovaná vstupem do bloku funkce (*prolog*) a automaticky uvolněna výstupem z tohoto bloku (*epilog*). Podílí se na ní registry ESP (*Stack Pointer*) ukazující na vrchol zásobníku a EBP (*Base Pointer*) ukazující na místo v zásobníku, kde začal blok. Alokace paměti se děje posuny těchto registrů.
- **dynamická**: proměnná je alokovaná i uvolněná příkazem programu (new, delete). OS musí alokovat souvislý blok paměti, k čemuž se používá datová struktura haldy (*heap*). Při alokaci se občas musí přesunout bloky paměti, aby se vytvořilo potřebné místo a je to tedy potenciálně relativně pomalá operace
- **statická**: tato paměť je narozdíl od dynamické alokovaná při spuštění a uvolněna při skončení programu, musí tedy být k dispozici při spuštění a nelze ji za běhu uvolnit