8. Řízení operační paměti (OP)

01 K čemu slouží operační paměť

02 Co se nachází v operační paměti

03 Strategie přidělování operační paměti

- a) Souvislá oblast
- b) Po blocích
- c) Stránkování
- d) Segmentace
- e) Segmentace se stránkováním

04 Výpadek stránky, pre-cleaning, thrashing

05 Čisté a špinavé stránky

06 Algoritmy výměny stránek

8. Řízení operační paměti

K čemu slouží operační paměť a co se v ní nachází?

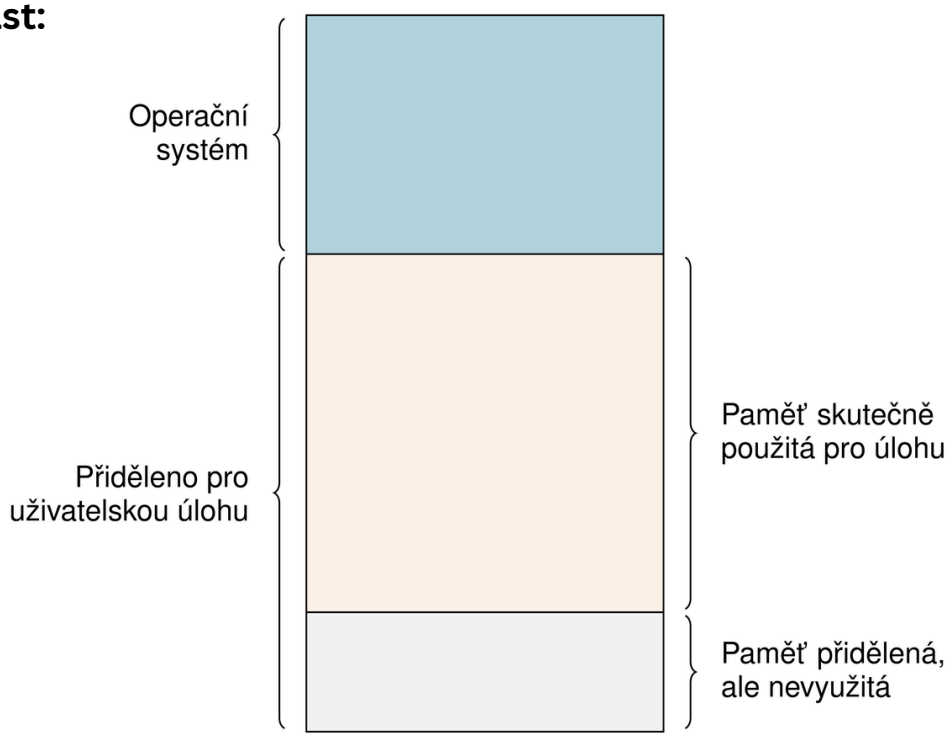
- **Operační paměť** *(RAM - Random Access Memory)* je zařízení pro **dočasné uchovávání dat a běžících programů**, které procesor aktuálně zpracovává.
- Patří mezi **RWM paměti** *(Read Write Memory)*, což znamená, že z ní lze data **číst i zapisovat.**
- Operační paměť je **volatilní** – její obsah je **závislý na napájení** a po odpojení od elektrické sítě dojde ke **ztrátě všech dat**.
- **Obsahuje:**
  - Běžící programy *(např. operační systém, aplikace, systémové procesy)*
  - Data, která jsou **aktuálně zpracovávána** *(např. výpočty, uživatelské vstupy, mezipaměť procesoru)*.
- Slouží jako **dočasné úložiště pro informace**, které jsou často potřeba, aby byly rychle dostupné procesoru.
- **RAM** je podstatně **rychlejší** než vnější paměťová média jako **HDD** nebo **SSD**, což umožňuje efektivní zpracování dat.
- Umožňuje rychlý přístup k libovolné části paměti **bez nutnosti sekvenčního čtení** *(na rozdíl od mechanických disků)*.

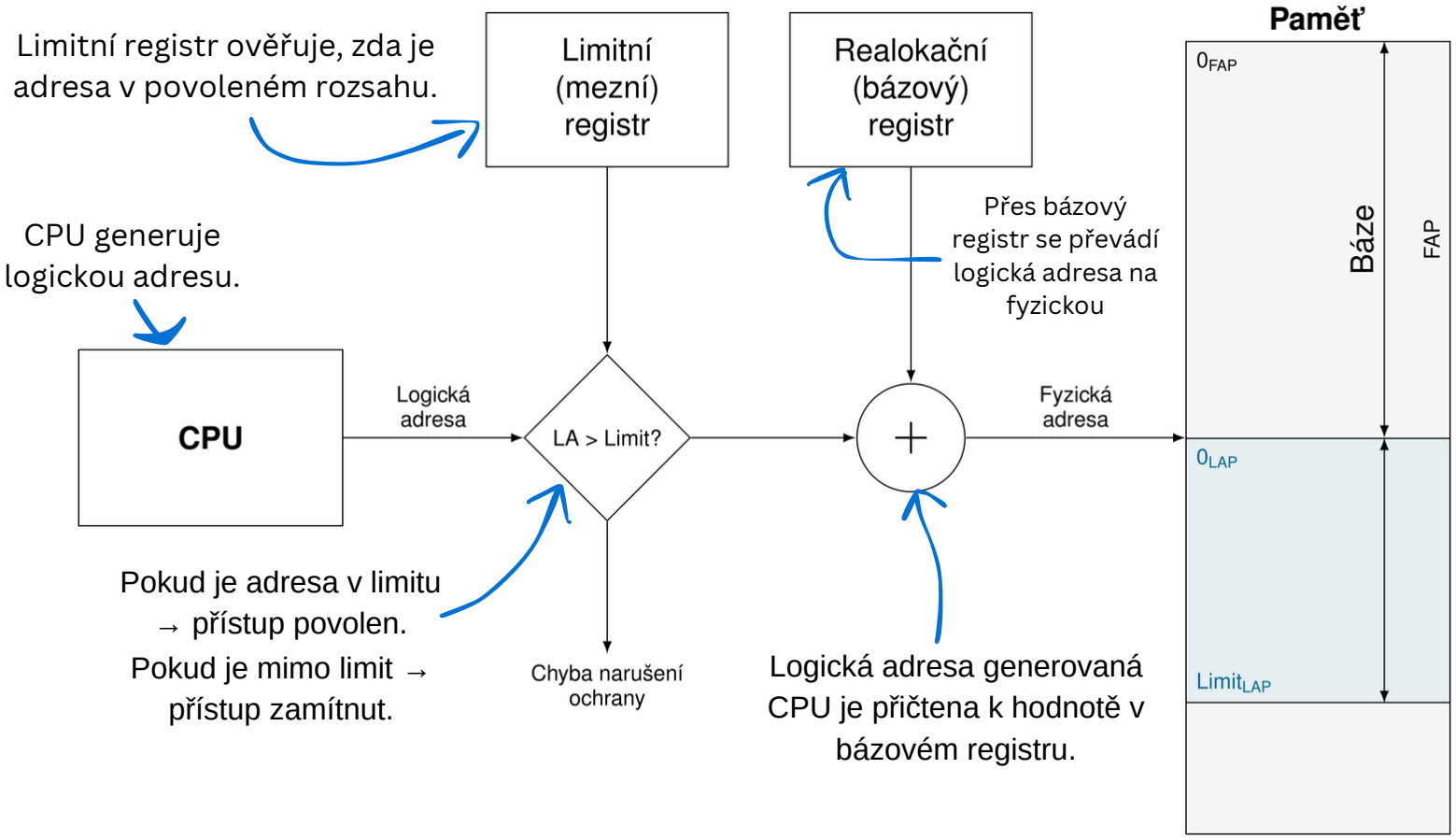
Strategie přidělování místa v paměti – přidělení veškeré volné paměti *(souvislá oblast)*

- Část operační paměti je **rezervována operačním systémem** *(OS)*, což zahrnuje **kód OS**, **vyrovnávací paměť** *(cache)* apod.
- Zbytek operační paměti je dostupný **uživatelským programům**.
- V daném okamžiku může být v paměti **aktivní pouze jeden uživatelský program**, což znamená, že **všechny prostředky jsou přiřazeny aktuálně běžícímu programu**.
- **Bázový registr:** Uchovává **počáteční adresu paměti**, kterou program používá
- **Logické adresy:** Adresy generované programem *(relativní k jeho začátku)* se přičítají k hodnotě bázového registru, čímž vznikne **fyzická adresa v paměti**.
- **Ochrana paměti** je zajištěna pomocí:
  - **Limitního registru:** Určuje **maximální velikost paměti**, kterou může program využívat.
  - Pokud program překročí hodnotu limitního registru, **přístup k paměti je zablokován**. Tím se zabraňuje přístupu programu mimo jemu přidělený prostor a **chrání se ostatní data** či procesy v paměti.
- **Bázový registr** má hodnotu **2000** *(začátek přidělené paměťové oblasti)*.
- **Limitní registr** má hodnotu **8000** *(konec přidělené paměťové oblasti)*.
- Proces požádá o přístup na logickou adresu 50. Výpočet fyzické adresy je:
  - **Fyzická adresa** = **2000** *(bázový registr)* **+ 50** *(logická adresa)* = **2050**.
    - Fyzická adresa je **porovnána** s limitním registrem *(8000)*.
    - Pokud je fyzická adresa **v rozmezí** *(2000–8000)*, proces pokračuje.
    - Pokud je **mimo tento rozsah**, přístup je **zablokován**.

8. Řízení operační paměti

Souvislá oblast:





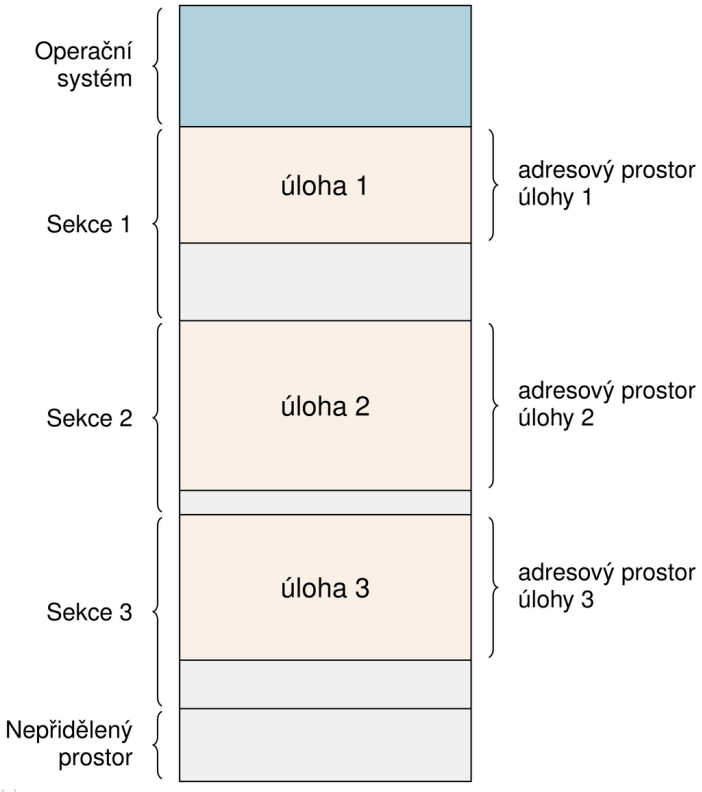
8. Řízení operační paměti

- **Swapping (odkládání procesů na úložiště):**
  - Pokud operační paměť **nemá dostatek prostoru** pro všechny procesy, využívá se **swapping**:
    - **Swapping:** Proces, při kterém je aktuální **program uložen na pevný disk** *(HDD nebo SSD)* a **nahrazen jiným procesem, který potřebuje paměť**.
    - Když je potřeba vrátit původní proces zpět do operační paměti, načte se zpět z úložiště.
- **Prokládání paměti *(segmentation):***
  - Pokud program **přesahuje velikost** dostupné paměti:
  - Paměť je rozdělena na **dvě části**:
    - **Fixní část:** Obsahuje **klíčové instrukce programu**, které musí být **trvale v paměti**.
    - **Překrývající část:** Část, která může být nahrazena jinými daty nebo programy dle potřeby.
  - Překryvání je **řízeno uživatelem** *(programem)*, nikoli operačním systémem.

Strategie přidělování místa v paměti – po blocích

- Na rozdíl od strategie přidělování celé volné paměti *(související oblast)*, kde může být v paměti v daném okamžiku pouze jeden proces, tato strategie **umožňuje běh více procesů současně.**
- Bloky **mohou růst**, ale **ne se zmenšit**.
- Paměť je **rozdělena na bloky** *(sekce)* o **stejné velikosti**. Bloky mohou být **obsazeny procesy** nebo mohou být **volné** *(tzv. „díry“)*.
- Před přidělením bloku musí **správce paměti znát velikost úlohy** *(procesu)*, aby bylo možné určit, **zda se proces vejde** do volného bloku.
- Pokud jsou **volné bloky vedle sebe**, správce paměti je může **sloučit do většího bloku**.
  - To je důležité, pokud je potřeba **alokovat větší proces**, který by se do jednotlivých malých volných bloků nevešel.
  - **Omezení:** Správce paměti dokáže sloučit **pouze sousedící** volné bloky.

Po blocích:

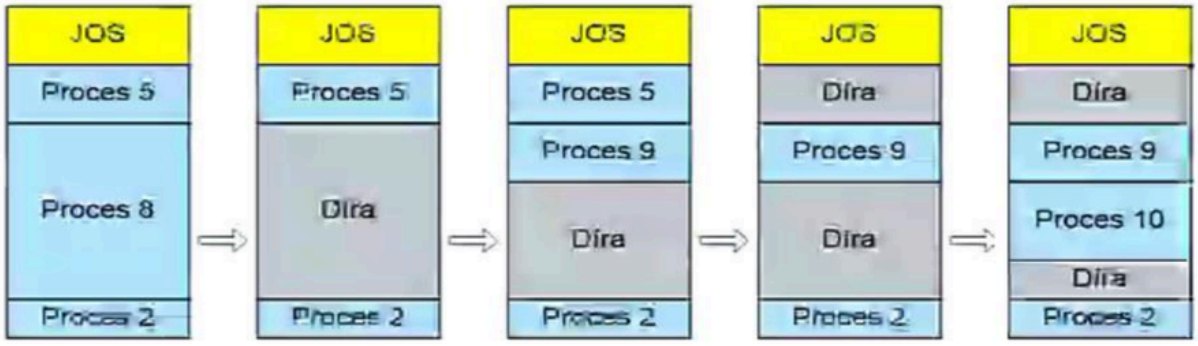


8. Řízení operační paměti

- V systému **MS DOS** byla tato strategie využívána, což reflektuje **jednoduchost implementace** a méně složité požadavky na správu paměti.
- **Výhody:**
  - Jednoduchá implementace, jelikož se bloky alokují a uvolňují relativně přímočaře.
- **Nevýhody:**
  - Dochází k značné **fragmentaci paměti**, což znamená, že volné paměťové bloky mohou být **rozprostřeny po celé paměti a nejsou spojité**.
  - Fragmentace zvyšuje náročnost správy paměti, protože je nutné monitorovat a slučovat volné bloky, což vyžaduje čas i výkon.
- **Defragmentace** paměti:
  - Fragmentaci lze odstranit procesem **defragmentace**, který **přeuspořádá paměťové bloky** tak, aby volné bloky **byly spojité**.
  - Tento proces se nazývá **setřásání paměti**. Správce paměti **přemístí obsazené bloky blíže k sobě** a uvolní souvislou oblast volné paměti.

Statické a dynamické přidělování sekcí

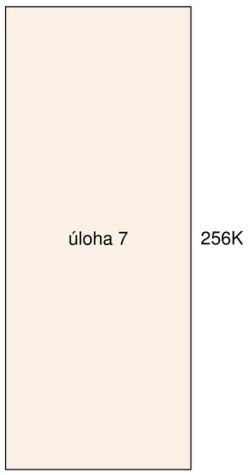
- **Statické přidělování sekcí:**
  - Paměť je rozdělena na **pevné sekce** *(bloky)* **předem**, bez ohledu na aktuální potřebu procesů.
  - Výhodou je **jednoduchost správy**, ale nevýhodou je **možné plýtvání pamětí**, pokud proces nevyužije celý blok.
- **Dynamické přidělování sekcí:**
  - **Paměťové bloky** *(sekce)* se vytvářejí **až při vzniku procesu**.
  - Správce paměti může **periodicky slučovat volné oblasti do jedné větší oblasti**, což **eliminuje fragmentaci** a umožňuje efektivnější využití paměti.



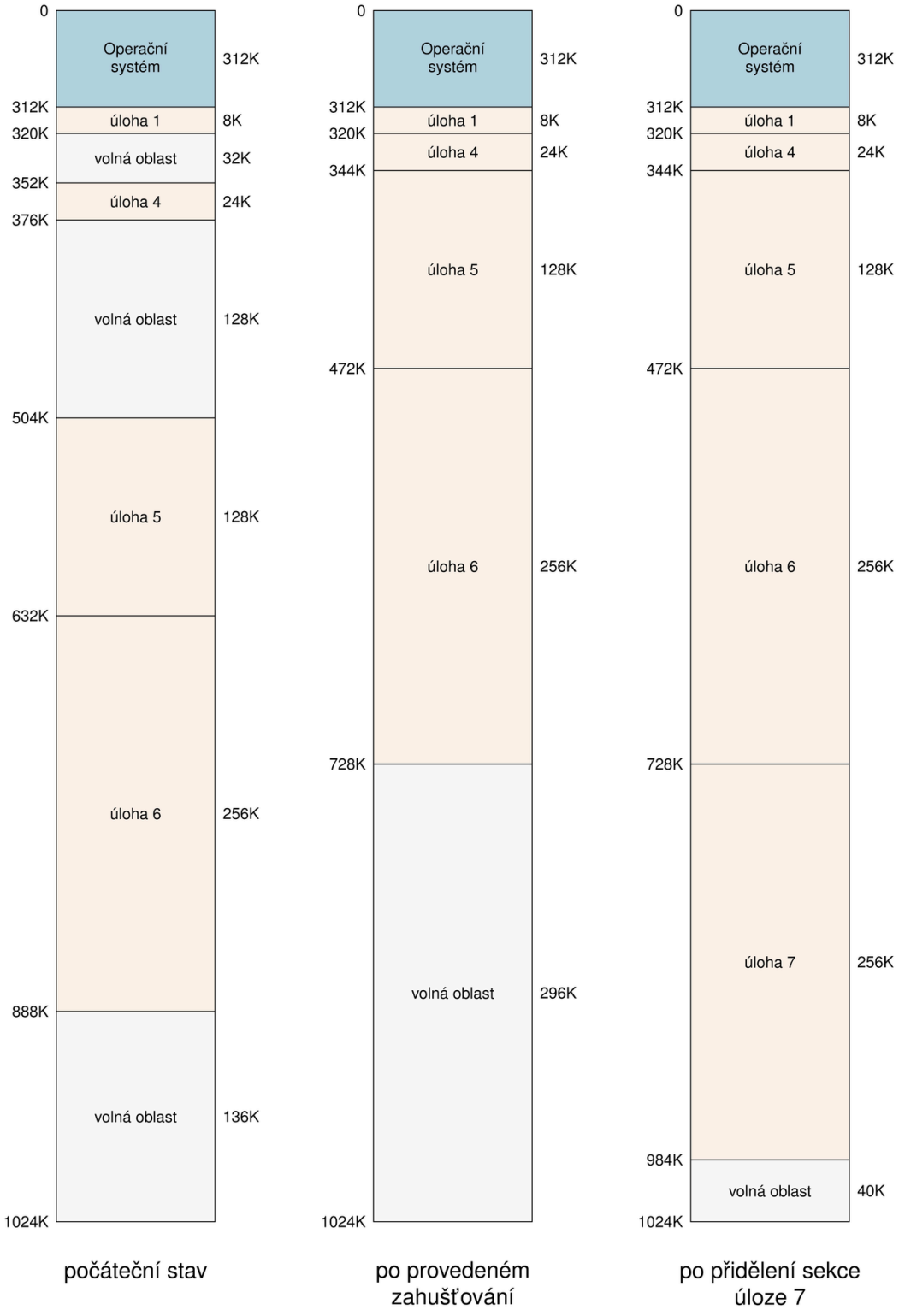
- Paměť je rozdělena na **bloky** *(sekce)*, které obsahují **procesy** nebo jsou **prázdné** *(„díry“)*.
- Po přidělení a ukončení procesů vznikají **díry**. Tyto volné bloky mohou být příliš malé na to, aby do nich šel vložit nový proces.
- **Řešení:** Správce paměti provede **setřásání paměti** *(defragmentaci)*, čímž se volné bloky přesunou vedle sebe a vytvoří **větší souvislý volný blok**.
- Zkratka **JOS** označuje **Job Operating System**: **spravuje alokaci paměti**.

8. Řízení operační paměti

Chceme do paměti přiřadit úlohu 7 (256K), na tu ale ze začátku není místo. Po zahuštění (setřásání paměti) jsme schopni pro úlohu místo (296 K) vytvořit a do paměti se tak vleze.



Dynamické přidělování sekcí (lepší obrázek):



8. Řízení operační paměti

Alokační strategie

- **FIRST FIT:**
  - Proces je umístěn do **prvního volného bloku**, který má dostatečnou velikost.
  - **Výhody:**
    - Nejjednodušší **implementace**.
    - Rychlé rozhodování.
  - **Nevýhody:**
    - Zanechává **fragmentaci** na začátku paměti.
- **BEST FIT:**
  - Proces je umístěn do **nejmenšího bloku**, do kterého se vejde.
  - **Výhody:**
    - **Minimalizuje** zbytkovou velikost volného bloku.
  - **Nevýhody:**
    - **Složitější** na implementaci, protože je nutné prohledat všechny volné bloky.
- **LAST FIT:**
  - Proces je umístěn do **posledního volného bloku**, který má dostatečnou velikost.
  - **Výhody:**
    - Umožňuje **efektivní využití** volných bloků na konci paměti.
  - **Nevýhody:**
    - Vyžaduje **průchod celé paměti**.
- **WORST FIT:**
  - Proces je umístěn do **největšího** volného bloku.
  - **Výhody:**
    - Zanechává větší volné bloky, což může **usnadnit přidělování velkých procesů.**
  - **Nevýhody:**
    - Může vést k **neefektivnímu využití paměti**.

8. Řízení operační paměti

Strategie přidělování místa v paměti – Stránkování (Paging)

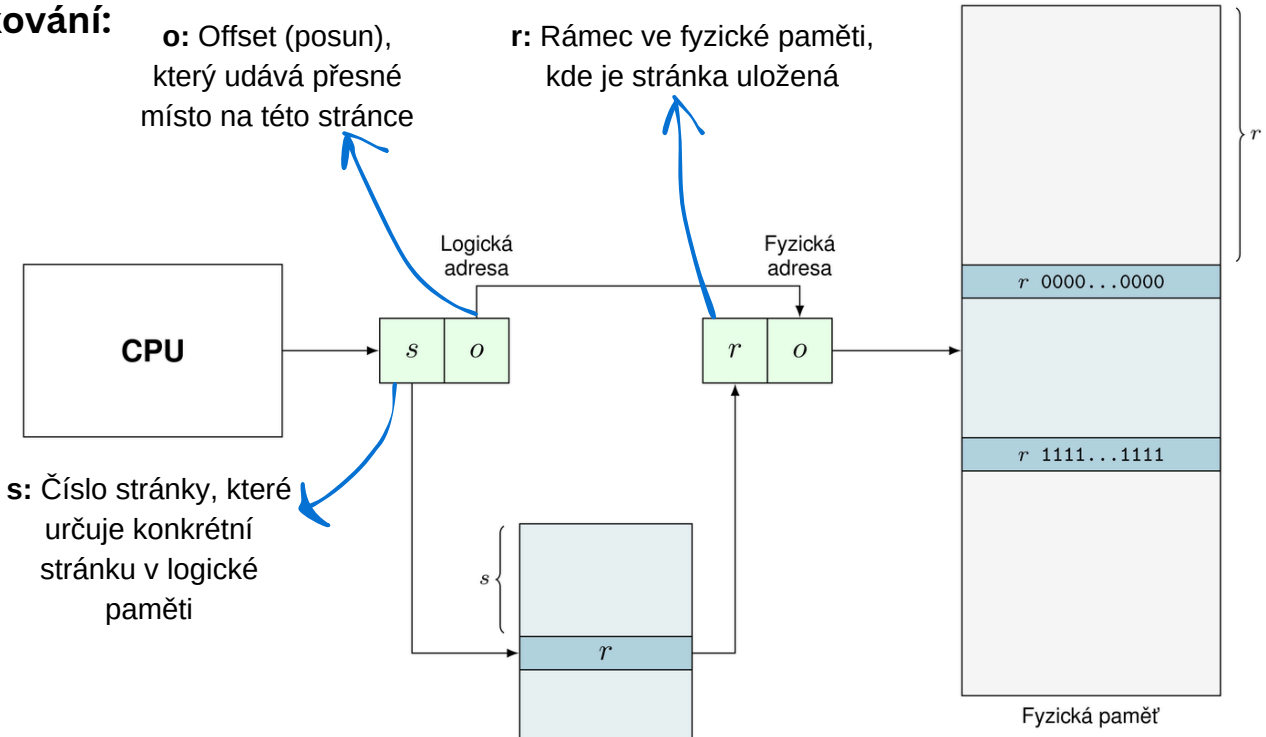
- **Logická paměť: Logická adresa**, kterou používají procesy, je rozdělena na **rámce** *(angl. frames)* stejné velikosti.
  - **Rámec** je **část paměti**, která je **pevné velikosti** = určitému počtu **bajtů** v reálné paměti.
- Proces je rozdělen na **stránky** *(pages)* o **stejné velikosti jako rámce**.
- Číslování stránek začíná vždy **od 0:** Procesor používá **stránkovou tabulku**, která **mapuje logické stránky na fyzické rámce**.
- **Offset:** Každá stránka má **uvnitř vlastní offset**, což je hodnota vyjadřující **pozici dat** na dané stránce.
- **Logická adresa se skládá z:**
  - **Čísla stránky** *(page number)*: Označuje **konkrétní stránku v paměti**.
  - **Offsetu** *(page offset)*: Určuje konkrétní pozici uvnitř dané stránky.
- Pokud je velikost stránky **4 KB**, offset potřebuje **12 bitů** *(protože 2^12 = **4096**)*.
- **Překlad logické adresy na fyzickou:**
  - Logická adresa je zadána do **tabulky stránek** *(page table)*.
    - Tabulka stránek obsahuje informace o tom, **kde se daná stránka nachází** v rámcích fyzické paměti.
  - Procesor z tabulky zjistí **odpovídající fyzický rámec**.
  - Přes **offset** se určí **konkrétní místo** v rámci.
- **Vlastnosti a výhody stránkování**
  - **Odstranění fragmentace:** Stránkování eliminuje externí fragmentaci, která vzniká u metod, kde paměť není rozdělena na pevné části.
  - **Flexibilní přidělování:** efektivní využití tím, že se přidělují pouze potřebné stránky.
  - **Sdílení paměti mezi procesy:** Několik procesů může sdílet části paměti tím, že se jejich logické adresy mapují na stejné fyzické rámce.
- **Nevýhody stránkování**
  - **Vnitřní fragmentace:** Poslední stránka procesu nemusí být zcela zaplněná, což vede k plýtvání místem.
  - **HW podpora:** Stránkování vyžaduje hardwarovou podporu, konkrétně procesor musí obsahovat jednotku pro správu paměti (MMU).

8. Řízení operační paměti

Stránkování:

o: Offset (posun), který udává přesné místo na této stránce

r: Rámec ve fyzické paměti, kde je stránka uložená

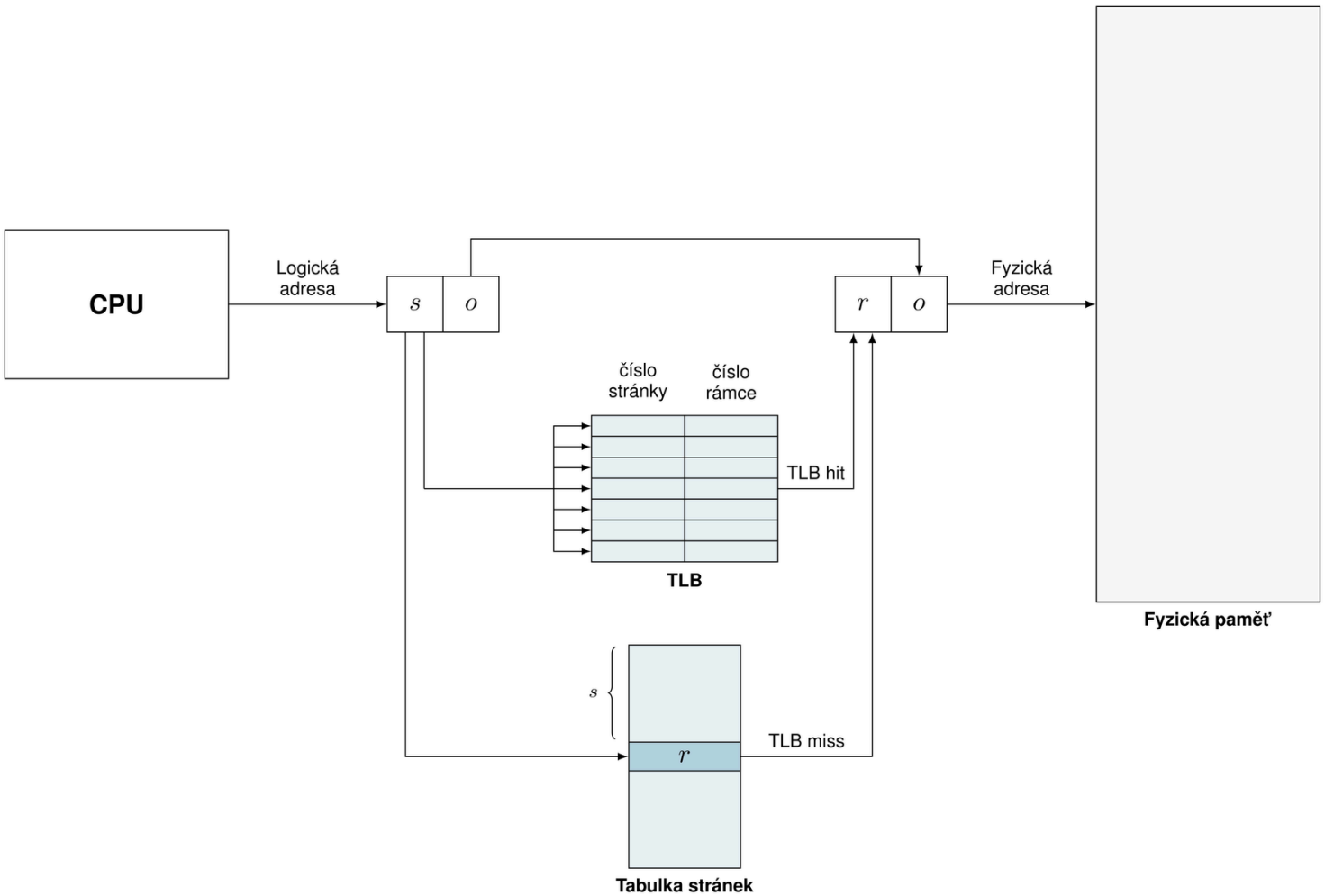


s: Číslo stránky, které určuje konkrétní stránku v logické paměti

Fyzická adresa vznikne kombinací čísla rámce **(f)** a offsetu **(d)**.

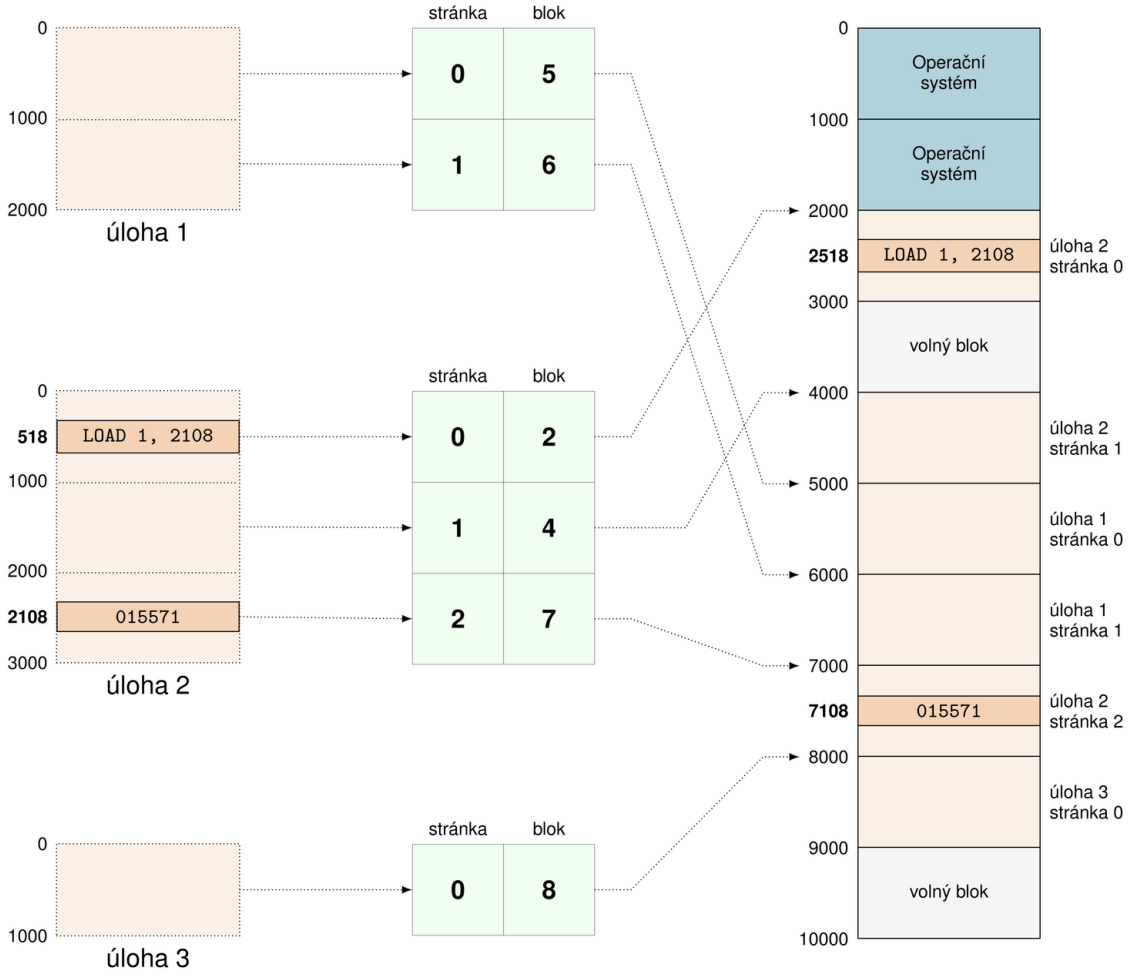
Tabulka převádí číslo stránky **(p)** na číslo rámce **(f)** ve fyzické paměti.

Například stránka s=3 může být uložena ve fyzickém rámci r=7.



8. Řízení operační paměti

1. **Logické stránky**: Úlohy jsou rozděleny na menší logické stránky.
2. **Tabulka stránek**: Mapování stránek na fyzické bloky je uloženo v tabulce stránek (zelené bloky).
3. **Fyzická paměť**: Fyzické bloky jsou nepravidelně rozděleny mezi úlohy podle jejich potřeb a dostupnosti paměti.
4. **Využití volné paměti**: Pokud není dostatek paměti, nevyužité bloky mohou být přiděleny dalším procesům.



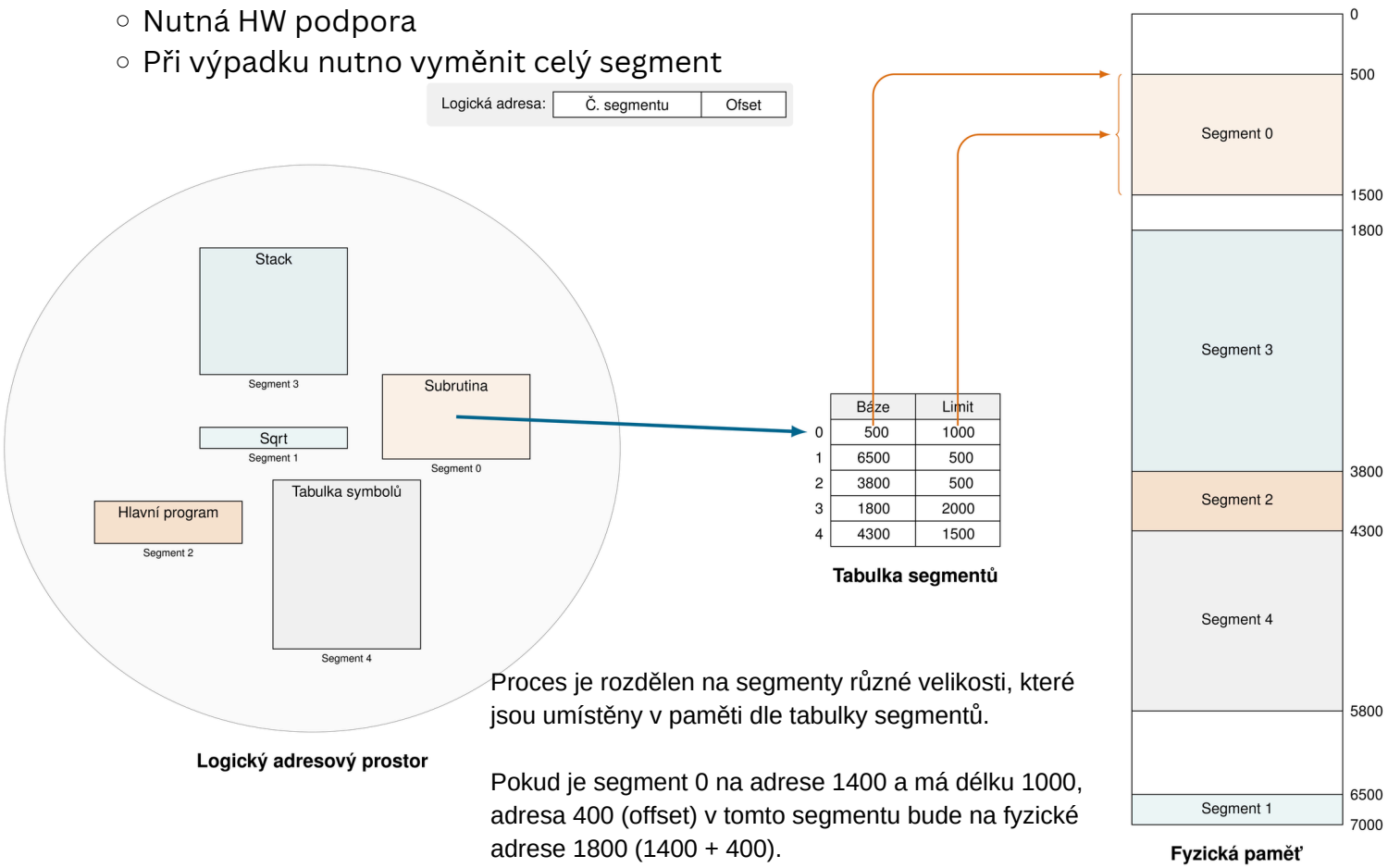
Stránkování na žádost

- Stránkování na žádost umožňuje procesům **využívat paměť efektivněji** tím, že se stránky načítají do paměti pouze **tehdy, když jsou potřeba**.
- Pokud **fyzická paměť nestačí**, některé stránky procesů jsou **odloženy na disk** do tzv. **swapovacího oddílu**.
- **Tabulka stránek** uchovává informace o tom, zda je stránka uložená **v paměti, nebo na disku**, a **adresu stránky na disku**.
- Když proces potřebuje stránku, která je na disku, dojde **k výpadku stránky** *(Page Fault)*.
- **Při výpadku stránky obslužný program zajistí:**
  - Načtení požadované stránky **z disku do paměti**.
  - Aktualizaci tabulky stránek, aby odrážela **aktuální stav**.
  - Opakování instrukce, která výpadek způsobila.
- Pokud je fyzická paměť plná, je třeba přesunout jinou stránku z paměti na disk podle **algoritmu pro náhradu stránek** *(např. FIFO, LRU, Optimal)*.
- Stránkování na žádost umožňuje **spouštění procesů větších, než je dostupná fyzická paměť, a efektivnější využití paměti**.
- Nevýhodou je **snížení výkonu systému** kvůli častému přenosu stránek mezi diskem a pamětí.
- **Vyžaduje podporu od procesoru a operačního systému**, aby bylo možné efektivně pracovat s výpadky stránek.

8. Řízení operační paměti

Strategii přidělování místa v paměti – segmentace:

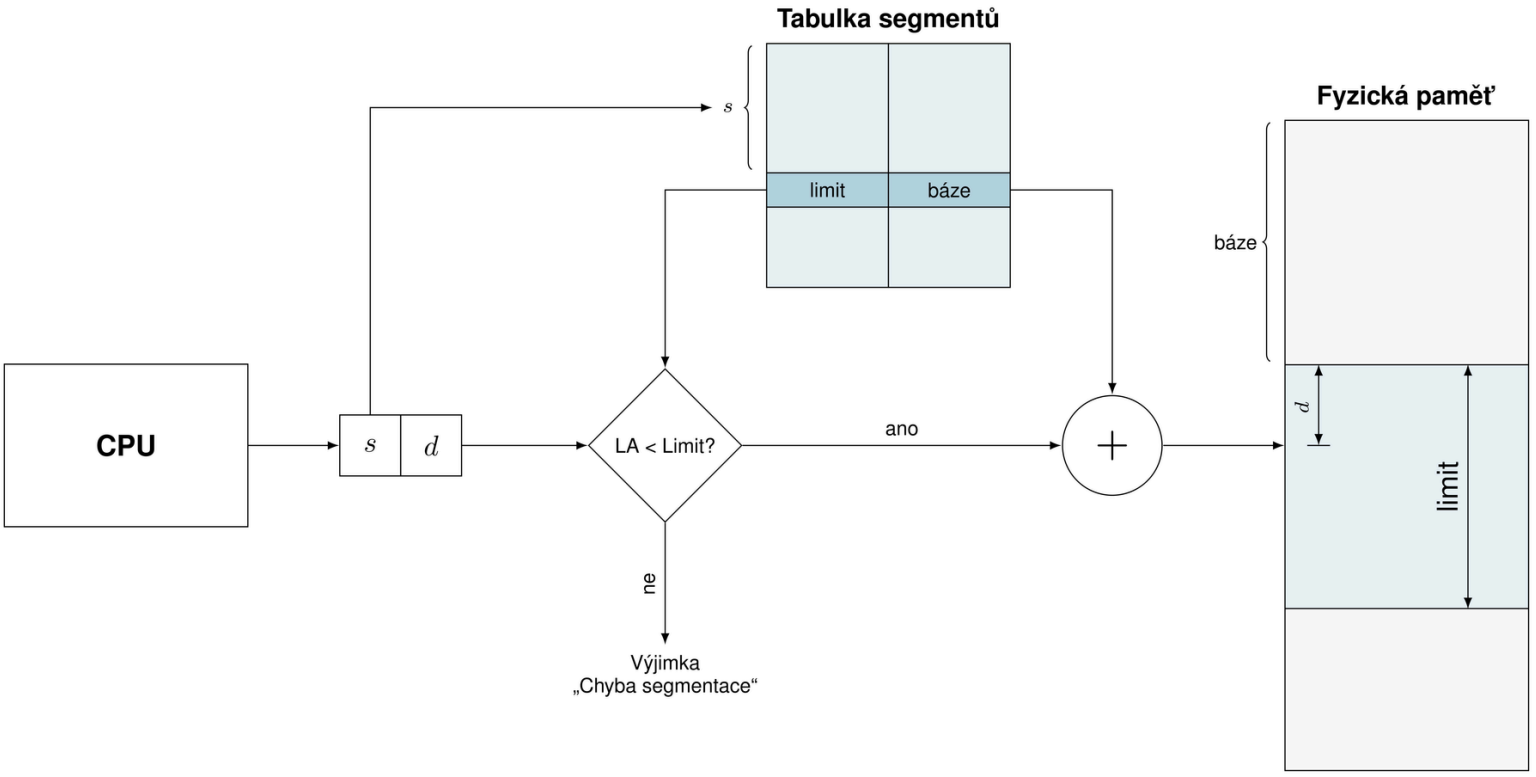
- Segmentace rozděluje program na **několik logických segmentů** *(např. hlavní program, zásobník, konstanty)*, které se **liší svou velikostí**.
- Každý **segment má své číslo** *(číslování od 0)* a proces je viděn jako logický celek.
- **Logická adresa se skládá z čísla segmentu a offsetu** *(vzdálenost od začátku segmentu)*. Offset musí být menší než velikost segmentu.
- **Tabulka segmentů:**
  - Obsahuje **bázovou adresu** (base) a limit pro každý segment.
  - **Base ukazuje začátek segmentu v operační paměti**, **limit jeho velikost**.
- Procesor **překládá logickou adresu na fyzickou adresu tak, že k bázové adrese přičte offset. Pokud offset překročí limit, dojde k chybě.**
- **Segmentace umožňuje:**
  - Sdílení kódu mezi procesy.
  - Lepší využití paměti díky eliminaci vnitřní fragmentace.
  - Detekci chyb, pokud je offset mimo limit segmentu.
- **Výhody:**
  - Segmenty mohou odpovídat skutečné potřebě jednotlivých částí programu.
  - Omezení vnitřní fragmentace.
  - Snadnější ladění programů díky odděleným segmentům.
  - Sdílení kódu mezi více procesy.
- **Nevýhody:**
  - Náročná alokace paměti, protože segmenty mají různé délky a nemusí se vejít do dostupného prostoru.
  - Možnost vzniku vnější fragmentace (volné místo mezi segmenty).
  - Nutná HW podpora
  - Při výpadku nutno vyměnit celý segment



Proces je rozdělen na segmenty různé velikosti, které jsou umístěny v paměti dle tabulky segmentů.

Pokud je segment 0 na adrese 1400 a má délku 1000, adresa 400 (offset) v tomto segmentu bude na fyzické adrese 1800 (1400 + 400).

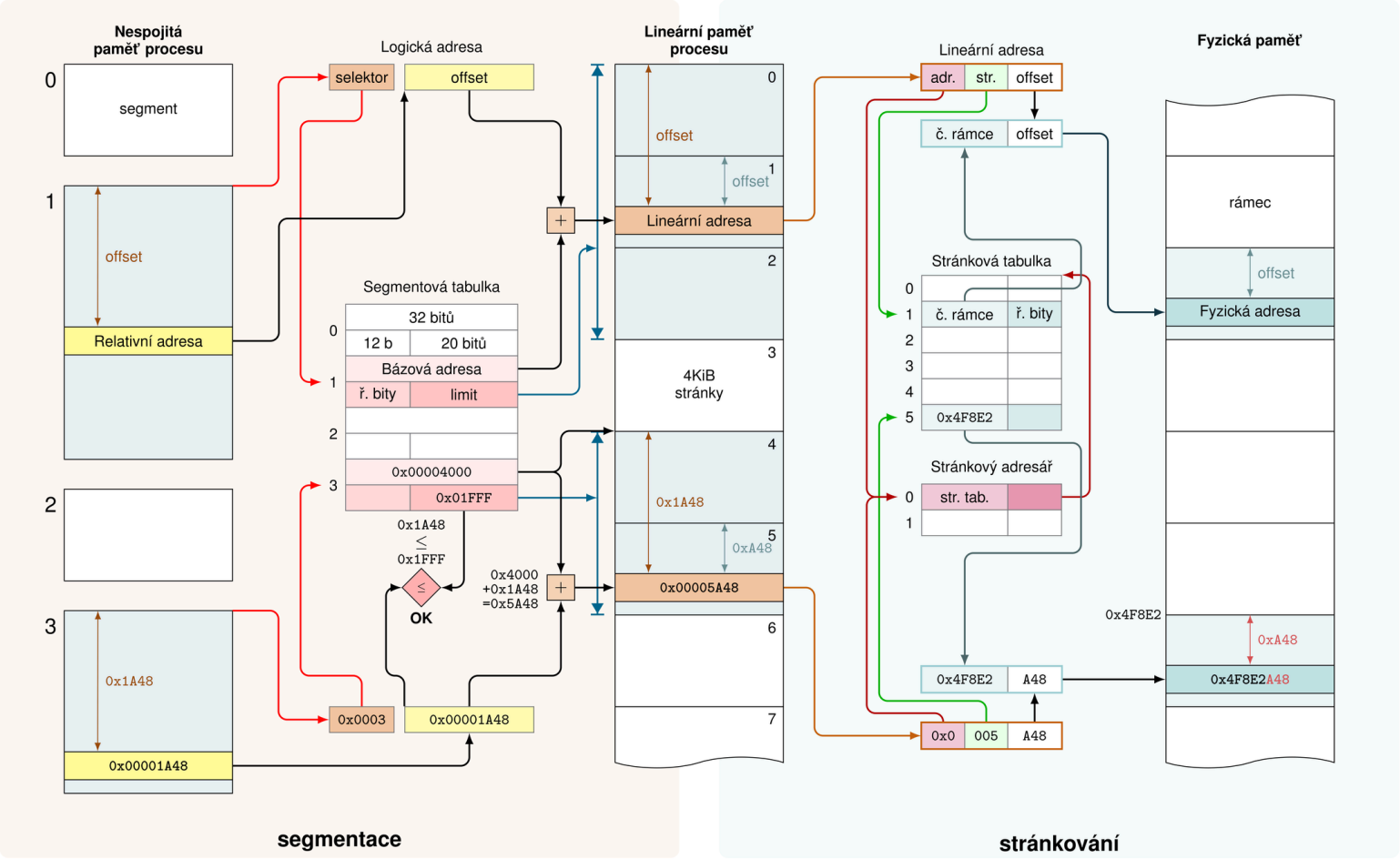
8. Řízení operační paměti



Strategie přidělování místa v paměti – segmentace se stránkováním

- Segmentace se stránkováním **kombinuje výhody obou přístupů:**
  - Program je rozdělen na **segmenty** *(hlavní program, zásobník, konstanty apod.)*
  - Každý **segment** je následně dále **rozdělen na stránky**, které jsou **uloženy v paměti jako rámce.**
- Velikost stránky/rámce je standardně **4 KB**.
- **Logická adresa** je **48bitová** a skládá se:
  - Z offsetu (12 bitů), který určuje posun v rámci stránky.
  - Ze selektoru (16 bitů) – zahrnuje index a přístupová práva.
  - Z dalších 3 bitů, které slouží pro práva a režim adresace.
- **Proces překladu adresy:**
  - **Selektor** určí **index** v tabulce segmentů a najde odpovídající položku obsahující **bázovou adresu a limit segmentu**.
  - **Limit** zajistí kontrolu, zda adresa **nepřekračuje velikost segmentu**.
  - Výsledná lineární adresa je rozdělena na **10bitové indexy** do stránkovací tabulky a **12bitový** offset.
  - Číslo stránky se vyhledá ve **stránkovací tabulce**, kde je uveden **rámec**.
  - Offset se **přičte k začátku rámce**, aby vznikla **fyzická adresa**.
- **Dva offsety:**
  - Jeden posun v rámci **segmentu**.
  - Druhý posun v rámci **stránky** *(menší, 12bitový)*.

8. Řízení operační paměti



Segmentace:

- Logická adresa obsahuje offset a selektor.
- Selektor určuje konkrétní segment v tabulce deskriptorů, kde se nachází:
  - Bázová adresa (začátek segmentu v paměti).
  - Limit, který omezuje maximální délku segmentu.
- Offset se přičítá k bázové adrese, čímž vzniká lineární adresa.

Stránkování:

- Lineární adresa je rozdělena na:
  - Indexy do víceúrovňové stránkovací tabulky (10 bitů pro každou úroveň).
  - Offset (12 bitů), který určuje posun v rámci rámce.
- Stránkovací tabulka převádí indexy na číslo rámce ve fyzické paměti.
- Fyzická adresa je vytvořena spojením rámce a offsetu.

8. Řízení operační paměti

Čisté a špinavé stránky:

- **Čisté stránky:**
  - Stránky, které nebyly modifikovány od svého načtení do paměti.
  - Není nutné je kopírovat zpět na disk, protože jsou identické s původní kopií uloženou na disku.
  - **Příklad:** Knihovny nebo statická data, která se nemění během běhu programu.
- **Špinavé stránky:**
  - Stránky, které byly změněny *(modifikovány)* během svého pobytu v operační paměti.
  - Musí být zálohovány *(zapsány zpět na disk)* před tím, než se uvolní z operační paměti, aby se data neztratila.

Čištění – Precleaning:

- Proces, kdy systém preventivně zálohuje data do virtuální paměti v době, kdy má volné prostředky.
- Tento přístup umožňuje optimalizaci a snižuje riziko ztráty dat.
- Pokud se stránka změní po zálohování, je nutné ji zálohovat znovu.

Výprask – Thrashing:

- **Výprask *(Thrashing)*:**
  - Problém nastává, když operační systém neustále přesouvá *(vyměňuje)* stránky mezi operační pamětí a virtuální pamětí na disku.
  - Tato situace výrazně snižuje výkon systému, protože procesor tráví většinu času správou paměti místo vykonávání procesů.
- **Řešení:**
  - Zvýšení kapacity operační paměti *(RAM)*.
  - Optimalizace algoritmu správy paměti.

Krátký výprask – Swap Storm:

- **Krátký výprask:**
  - Situace, kdy dojde k výraznému zpomalení systému kvůli nedostatku operační paměti.
  - Většinou se jedná o dočasný problém způsobený přetížením systému *(příliš mnoho aplikací najednou)*.
- **Řešení:**
  - Zavedení lepšího algoritmu správy paměti.
  - Uzavření některých aplikací, aby se uvolnily systémové zdroje.

Operační systémy a správa paměti:

- **Windows:**
  - Windows 7 a XP používají stránkování na žádost *(demand paging)*.
- **Linux:**
  - Linux využívá segmentaci nebo stránkování na žádost.
  - Často používá algoritmus NRU *(Not Recently Used)*, který označuje stránky dle jejich posledního použití a priorit.
- **Mac OS:**
  - Mac OS využívá stránkování na žádost, které zajišťuje efektivní správu paměti podle potřeby procesů.

8. Řízení operační paměti

Přehled algoritmů pro správu paměti (určeny pro stránkování)

Optimální algoritmus

- Nahrazuje stránku, která bude v budoucnosti volána ze všech nejpozději.
- Teoreticky nejefektivnější algoritmus.
- Není možné jej realizovat v praxi, protože nevíme, jaké stránky budou požadovány v budoucnosti.
- Slouží jako benchmark pro porovnání účinnosti jiných algoritmů.

FIFO (First In, First Out)

- Stránky jsou spravovány ve frontě.
- První stránka, která byla načtena, je první odstraněna.
- Může způsobit problém zvaný „anomálie Belady“, kdy přidání více rámců paměti může zvýšit počet výměn stránek.
- Jednoduchá implementace, ale nízká efektivita.

LRU (Least Recently Used)

- Nahrazuje stránku, která byla nejméně používána v minulosti.
- Sleduje čas posledního použití jednotlivých stránek.
- Bere v úvahu historické chování procesu.
- Vyžaduje dodatečnou paměť pro ukládání časových informací.

Druhá šance (Second Chance)

- Vylepšený FIFO algoritmus.
- Používá referenční bit k rozhodnutí, zda stránka dostane „druhou šanci“.
- Stránky jsou organizovány v kruhovém seznamu.
- Pokud stránka na začátku seznamu nemá nastavený referenční bit, je odstraněna.
- Pokud referenční bit nastaven je, stránka dostane druhou šanci a přesune se na konec seznamu.
- Zabraňuje odstranění často používaných stránek.

Hodinový algoritmus

- Modifikace algoritmu Druhá šance.
- Používá ručičku, která ukazuje na aktuální stránku v kruhovém seznamu.
- Ručička kontroluje referenční bit.
- Pokud není referenční bit nastaven, stránka je nahrazena.
- Pokud referenční bit nastaven je, ručička se posune k další stránce.
- Efektivní pro systémy s omezenými zdroji.

Random (Náhodný algoritmus)

- Vybere a nahradí stránku náhodně.
- Jednoduchá implementace.
- Neefektivní, nevychází z žádných historických nebo aktuálních dat.

NFU (Not Frequently Used)

- Každá stránka má čítač, který je pravidelně navyšován, pokud je stránka použita.
- Nahrazuje stránku s nejnižší hodnotou čítače.
- Bere v úvahu četnost používání stránek.
- Může nesprávně favorizovat stránky, které byly často používány v minulosti, ale aktuálně už nejsou.

8. Řízení operační paměti

NUR (Not Used Recently)

- Sleduje, zda stránka byla nedávno používána, pomocí dvou modifikátorů:
- **Referenční bit:** Označuje, zda byla stránka čtena/zapsána.
- **Modifikační bit:** Označuje, zda byla stránka změněna.
- Stránky jsou klasifikovány podle těchto bitů a prioritně jsou odstraněny stránky, které nejsou nedávno použity ani modifikovány.

Konkrétní příklady

Optimální algoritmus (7 výpadků)

požadavek	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	3	3	3
2		2	2	2	2	2	2	2	2	2	4	4
3			3	4	4	4	5	5	5	5	5	5

FIFO (9 výpadků)

požadavek	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	4	4	4	5	5	5	5	5	5
2		2	2	2	1	1	1	1	1	3	3	3
3			3	3	3	2	2	2	2	2	4	4

LRU (10 výpadků)

požadavek	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	4	4	4	5	5	5	3	3	3
2		2	2	2	1	1	1	1	1	1	4	4
3			3	3	3	2	2	2	2	2	2	5

Druhá šance (8 výpadků)

požadavek	1	2	3	1	4	2	5	1	2	5	1	4
1	1	1	1	1*	1	1	5	5	5	5*	5*	5
2		2	2	2	4	4	4	1	1	1	1*	1
3			3	3	3	2	2	2	2*	2*	2*	4