

Operační systémy

Přednáška 7: Správa paměti I

Správa paměti (SP)

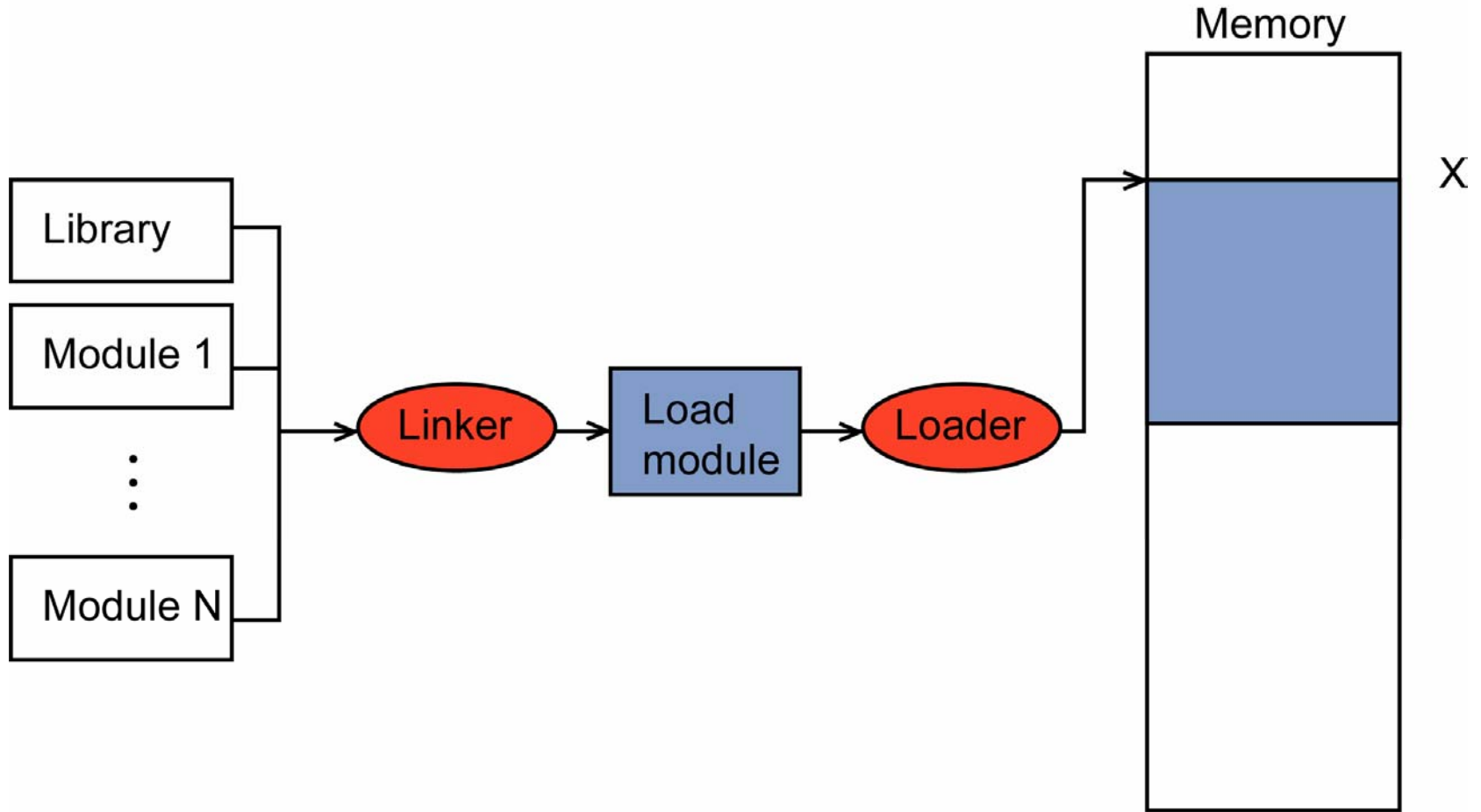
- **Memory Management Unit (MMU)**
 - hardware umístěný na CPU čipu
 - např. překládá logické adresy na fyzické adresy,...
- **Memory Manager**
 - software, který je součástí OS
 - udržuje **informaci volné a přidělené** paměti
 - **přiděluje a uvolňuje** paměť
 - zajišťuje **odkládání (swapping)** procesů

Požadavky na SP

- **Přemístění (Relocation)**

- při kompilaci není většinou známo umístění procesu ve fyzické paměti
- každý **odkaz do paměti** v programu **musí být přepočítán** podle aktuálního umístění procesu ve fyzické paměti
- v programu se můžeme **odkazovat na další programy**

Spojování a zavedení programu



Zavádění programu (loading)

- **absolutní zavedení (absolute loading)**

- každý odkaz do paměti v programu obsahuje absolutní fyzickou adresu
- program musí být zaveden vždy od dané fyzické adresy
- při sestavování programu musíme určit kam bude program zaveden

- **přemístitelné zavedení (relocatable loading)**

- každý odkaz do paměti v programu obsahuje relativní adresu (adresa vztažená k určitému bodu)
- informace o paměťových odkazech je uložena v „relocation dictionary“
- přepočítání relativní adresy na fyzickou se provede při zavedení programu do fyzické paměti

- **dynamic run-time loading**

- každý odkaz do paměti v programu obsahuje relativní adresu (adresa vztažená k určitému bodu)
- program je zaveden do paměti s relativními adresami
- relativní adresa se přepočte na fyzickou teprve při provádění instrukce

Spojování programu (linking)

- **statické spojování (static linking)**
 - vytvoří se jeden „load module“ s relativními adresami vztaženými k začátku modulu
- **dynamické spojování**
 - „load module“ obsahuje odkazy na další programy
 - **load-time dynamic linking**
 - odkazy na další programy se nahradí v okamžiku zavedení do paměti
 - **run-time dynamic linking**
 - odkazy na další programy se nahradí v okamžiku provádění instrukce

Požadavky na SP (2)

- **Ochrana**
 - každý proces musí být chráněn před nechtěnými přístupy z ostatních procesů
- **Sdílení**
 - umožnění přístupu ke společné paměti (např. několik procesů provádí stejný program)
- **Logická organizace (logický adresový prostor)**
 - lineární (jednorozměrný) adresový prostor
 - SW se obvykle skládá z modulů \Rightarrow více lineárních prostorů (např. s různými právy, kompilovány v různém čase, ...)
- **Fyzická organizace (fyzický adresový prostor)**
 - fyzická paměť se skládá z různých úrovní

Hierarchie pamětí

- **Inboard memory**

	velikost	čas přístupu	spravuje
registry	<1 KB	~1 ns	Compiler
L1 cache	1 MB	~10 ns	Hardware
L2 cache	0.5-8 MB	~20 ns	Hardware
hlavní paměť	16MB - 64 GB	~200 ns	Software

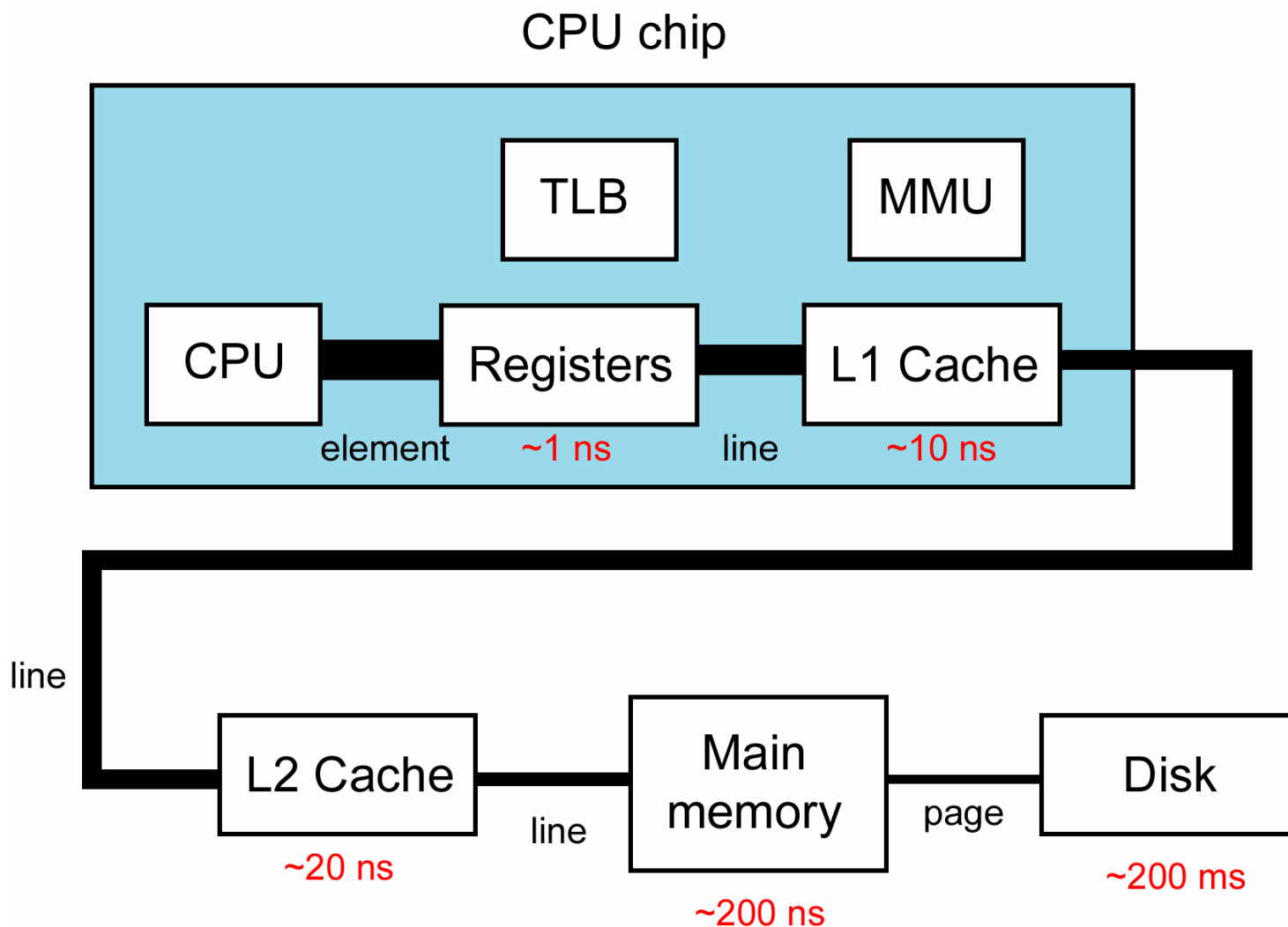
- **Outboard storage**

- magnetické disky 1G -11TB, 200ms, Software
- flash drive
- CD-ROM, CD-RW
- DVD

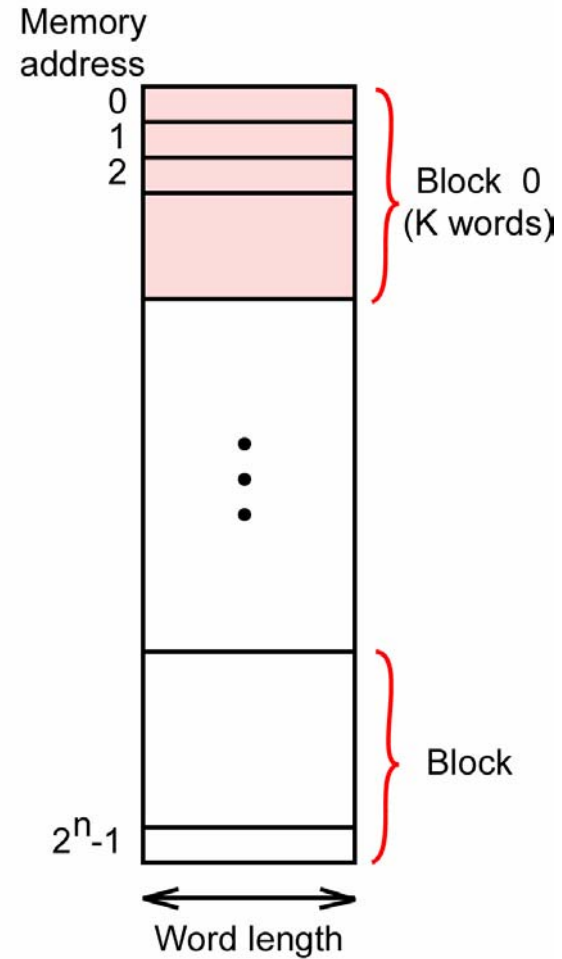
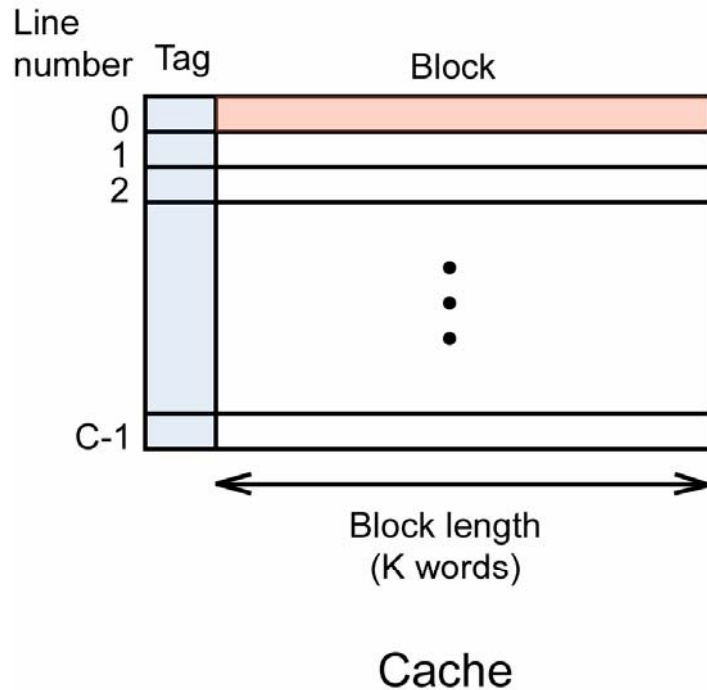
- **Off-line storage**

- magnetické pásky 20-100GB, 100s, Software

Hierarchie pamětí (2)



Cache – Hlavní paměť



Main memory

Cache design

- Velikost cache (cache size)
- Velikost bloku (block size)
- Mapovací funkce (mapping function)
- Nahrazovací algoritmus (replacement alg.)
- Strategie zapisování (write policy)

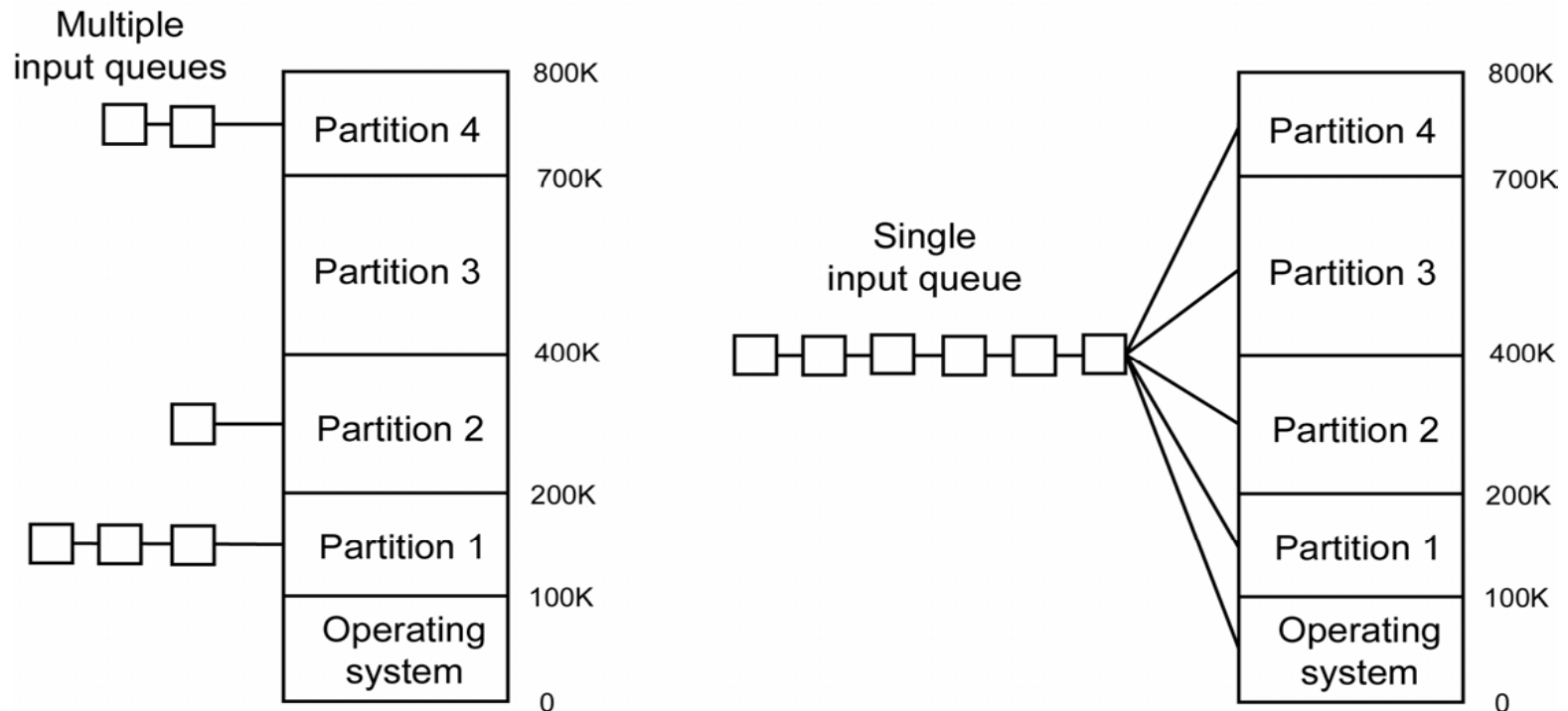
Základní techniky SP

- Fixní oblasti (fixed partitioning)
- Dynamické oblasti (dynamic partitioning)
- Jednoduché stránkování (simple paging)
- Jednoduchá segmentace (simple segmentation)

- Virtuální paměť se stránkováním (virtual memory paging)
- Virtuální paměť se segmentací (virtual memory segmentation)

Fixní oblasti

- **Paměť je rozdělena** na n oblastí různých velikostí (většinou při spuštění systému).
- Program je nahrán do stejně velké oblasti nebo větší.
- **Velkost paměťových oblastí se nemění** během běhu OS.



Fixní oblasti (2)

- **Oddělené vstupní fronty** pro každou oblast
 - **Nevýhoda**: nerovnoměrné obsazení front
- **Společná vstupní fronta**
 - Strategie výběru úlohy pro volnou oblast
 - **best fit** nalezení největší úlohy, která se vejde do oblasti
 - **first fit** nalezení první úlohy, která se vejde do oblasti
 - **Nevýhoda**
 - **best fit** znevýhodňuje malé (interaktivní) úlohy
 - **first fit** plýtvá místem velkých oblastí
 - **Řešení nevýhody best fit**
 - Úloha, která by mohla běžet, nesmí být předběhnuta více než k krát..
 - Při každém předběhnutí získá bod.
 - Pokud má úloha k bodů nemůže být předběhnuta.

Fixní oblasti (3)

- **Výhody**

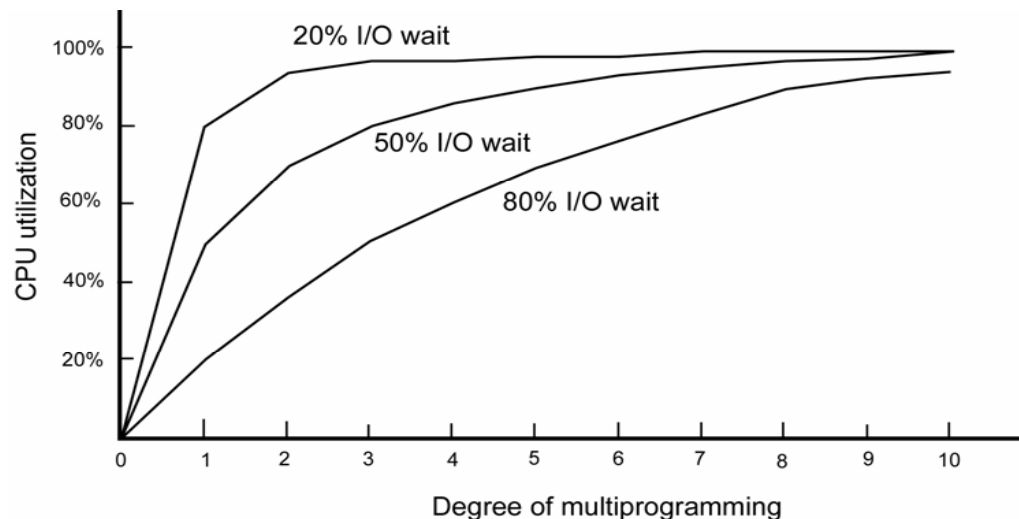
- jednoduchá implementace
- malé režijní náklady

- **Nevýhody**

- **interní fragmentace** (místo uvnitř oblasti není využito na 100 %)
- počet aktivních procesů je fixní

Modelování multiprogramování

- Multiprogramování zlepšuje využití CPU!
- **Pravděpodobnostní model využití CPU:**
 - Proces stráví zlomek svého času p čekáním na V/V.
 - Pokud máme současně n procesů v paměti, potom pravděpodobnost, že všechny procesy současně čekají na V/V je p^n .
 - **Využití CPU je $1 - p^n$.**
- Pravděpodobnostní model je **pouze přibližný odhad** (procesy nejsou na sobě nezávislé).



Přemístění a ochrana

- **Přemístění**

- adresy proměnných, návěští skoků,... jsou **přepočítány v okamžiku nahrání** programu do paměti
- sestavovací program musí do binárního programu zapsat informaci, která slova v programu obsahují adresy paměti, aby mohly být přepočítány

- **Ochrana**

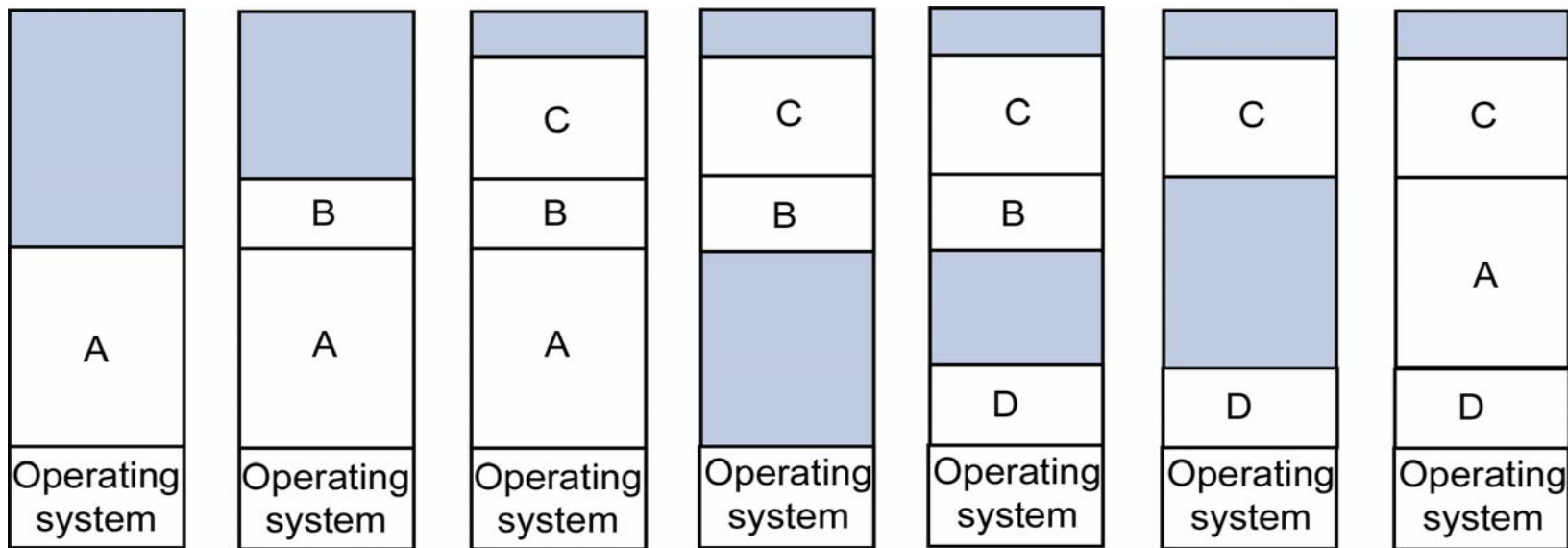
- paměť je rozdělena na bloky
- každý blok je svázán s n -bitovým **ochranným klíčem**, který určuje zda daná úloha smí přistupovat k datům v tomto bloku (IBM 360 - 2kB bloky, 4bitový klíč).

Přemístění a ochrana (2)

- Každá oblast paměti má dva registry (**bázový** a **limitní registr**), které obsahují nejmenší a největší adresu této oblasti.
- Když přistupujeme k paměti, bázový registr je přičten k adrese paměti a výsledek je porovnán s limitním registrem.
- **Nevýhoda:** potřeba sčítání a porovnávání při každém přístupu do paměti.
- **Řešení:** speciální HW.

Dynamické oblasti

- Počet, umístění a velikost oblastí se mění dynamicky,** tak jak jednotlivé procesy vznikají, zanikají a přesouvají se mezi hlavní pamětí a diskem.



Dynamické oblasti (2)

- **Výhody**

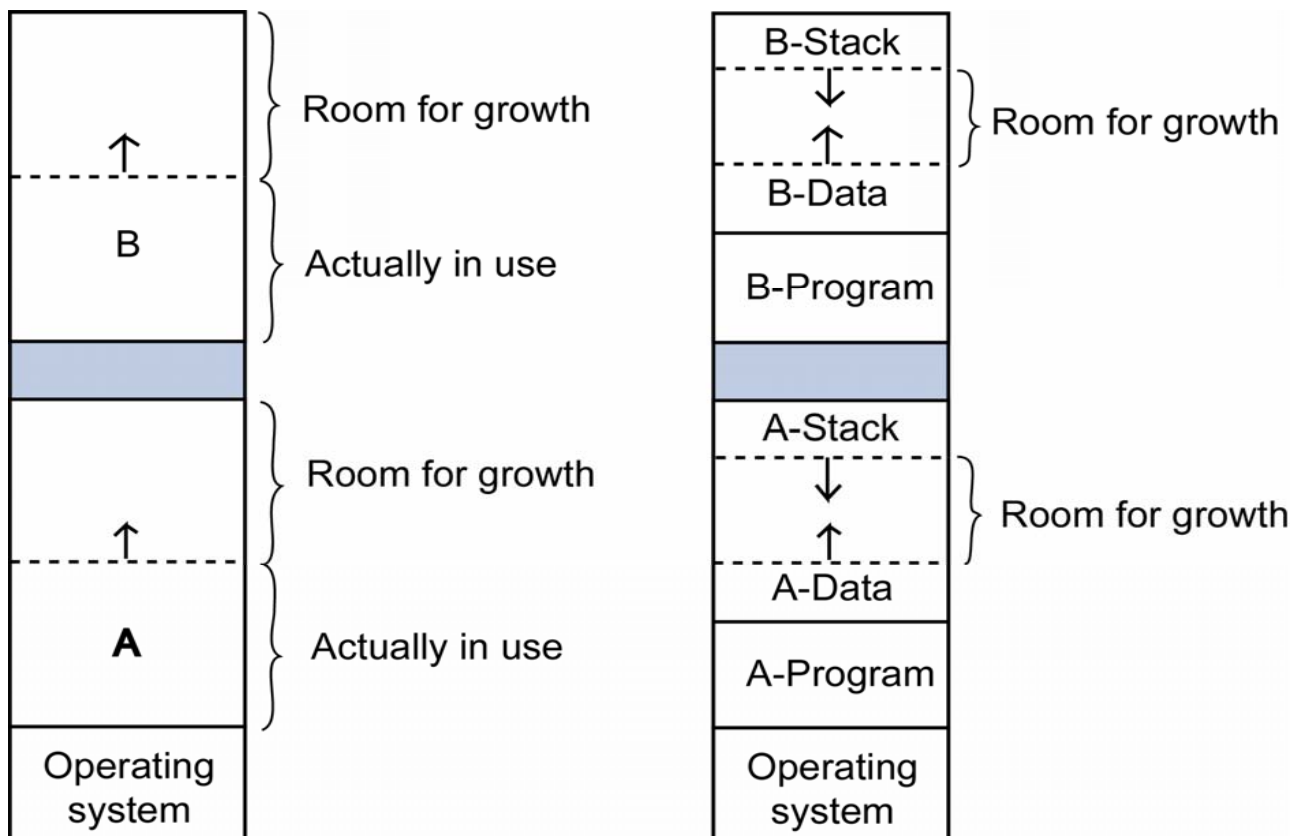
- „žádná“ interní fragmentace
- efektivnější využití paměti

- **Nevýhody**

- **externí fragmentace** (možno setřásání paměti, ale je to časově náročné)

Zvětšující se procesy

- **Datový segment** procesu **může měnit svojí velikost** během výpočtu.
- Proto musíme **alokovat více paměti** než je na počátku potřeba.

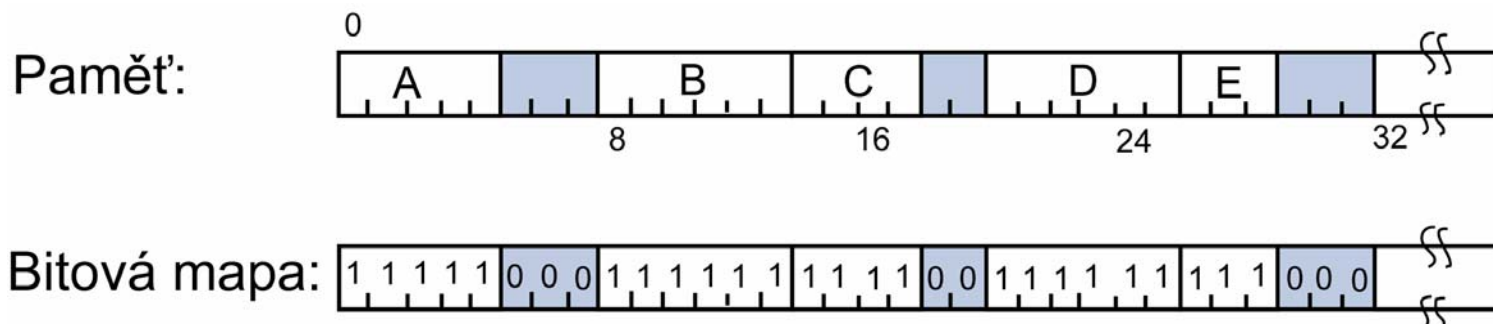


Správa použité a volné paměti

- **Jak udržovat informaci o volné a přidělené paměti?**
 - Bitové mapy.
 - Zřetězené seznamy.

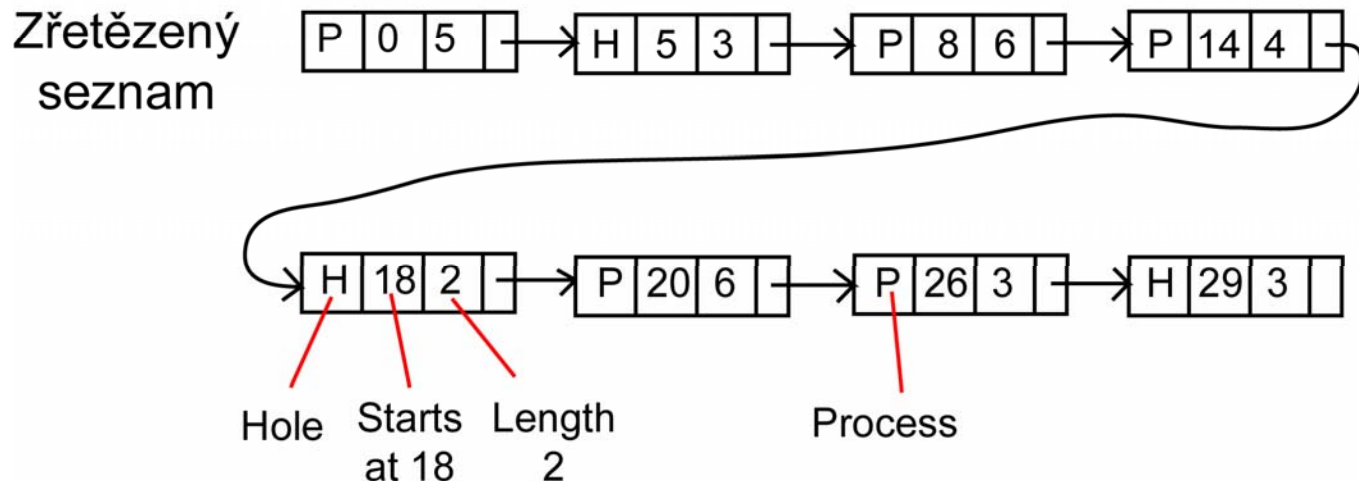
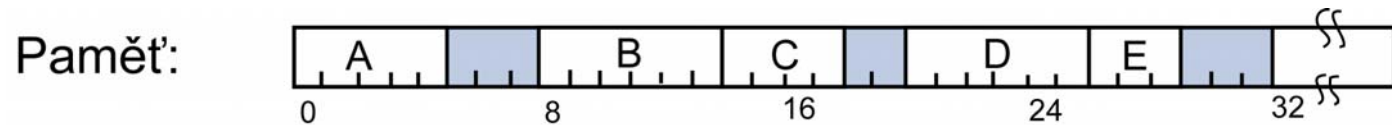
Bitové mapy

- Paměť je rozdělena na **alokační jednotky** (AU, veliké několik KB).
- Každá AU má korespondující bit ve speciálním řetězci—**bitové mapě** (0-volná, 1-přidělená).
- Nalezení volných AU = nalezení **souvislého řetězce** nulových bitů.
- Problémy:
 - **Velké AU** – malá bitová mapa x plýtvání hlavní paměti.
 - **Malé AU** – velká bitová mapa x lepší využití hlavní paměti.
 - **Hledání v bitové mapě je pomalé!**



Zřetěžené seznamy

- Zřetěžený seznam volných a přidělených paměťových segmentů (např. proces (P) a díra (H)).
- Seznam je **setříděn podle adres** segmentů.
- Když proces končí nebo je odložen na disk, aktualizace seznamu je jednoduchá.



Zřetězené seznamy (2)

- Paměť pro nový nebo odložený proces se může alokovat několika způsoby:
 - **first fit**
 - Nalezení první dostatečné díry od začátku seznamu.
 - Stávající díra se rozdělí na proces a díru.
 - **next fit**
 - Jako first fit, ale hledání začíná z místa, kde jsme skončili posledně.
 - Díry ze začátku seznamu nejsou preferovány.
 - **best fit**
 - Nalezení nejmenší díry, do které se daný proces vejde.
 - Stávající díra se rozdělí na proces a malou díru.
 - **worst fit**
 - Nalezení největší díry.
 - Nově zniklá díra bude dostatečné veliká pro další procesy.

Zřetězené seznamy (3)

- Na základě simulací, **next fit** vykazuje horší chování než **first fit**.
- **Best fit** je pomalejší než **first fit**, protože musíme prohledávat celý seznam.
- **Best fit** ve výsledku plýtvá pamětí více než **first fit**, protože paměť bude obsahovat velké množství malých děr.
- **Oddělené seznamy** pro procesy a díry
 - rychlejší alokace paměti
 - při uvolnění paměti se **sousední díry spojují**
 - vhodné použít **obousměrně zřetěžený seznam**
 - **nevýhoda**: složitější a tím i pomalejší operace uvolnění paměti.

Zřetězené seznamy (4)

- **quick fit**

- informace o dírách je udržována v několika oddělených seznamech,
- každý seznam obsahuje informaci o dírách jejichž velikost je v určitém intervalu
- např.:

seznam děr od 1 kB do 5 kB

seznam děr od 5 kB do 10 kB

seznam děr od 10 kB do 50 kB

...

- **Výhoda**

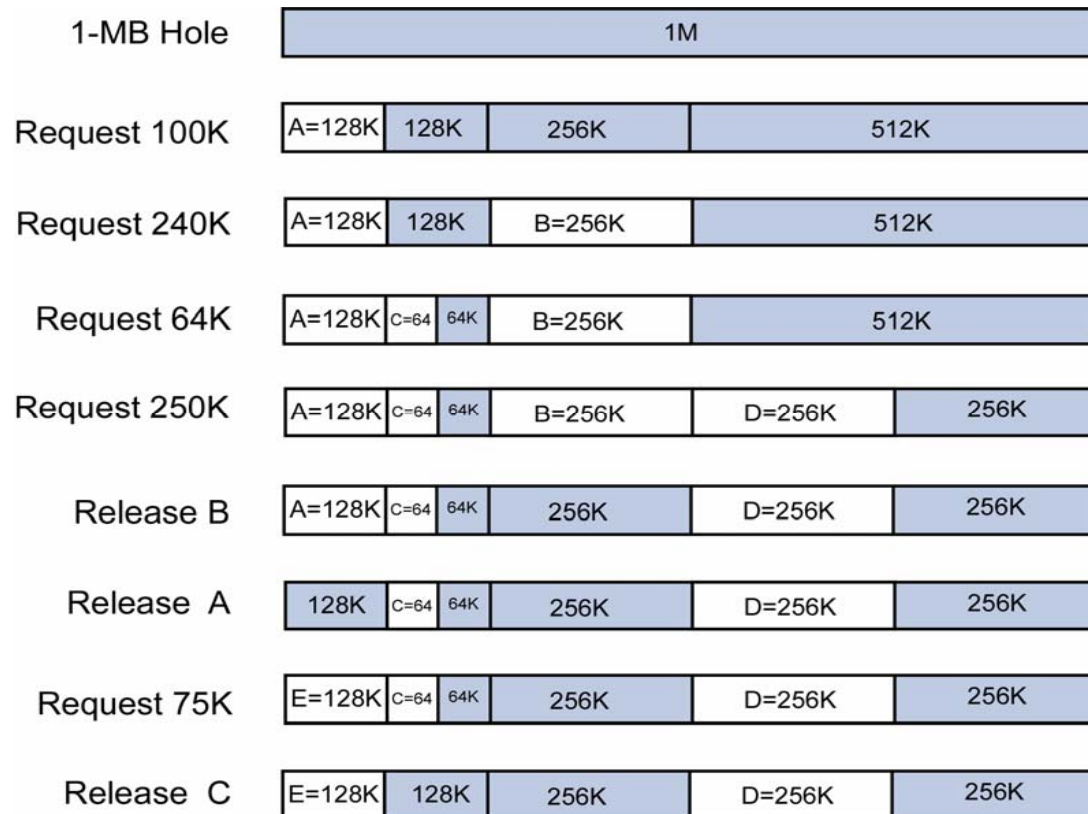
- Rychlé nalezení volné díry.

- **Nevýhoda**

- Nalezení sousedních děr pro sloučení do větší díry je časově náročné.

Zřetěžené seznamy (5)

- **Buddy systém = quick fit s dírami o velikosti 2^n bytů.**
- Informace o dírách je v několika oddělených seznamech pro velikosti děr 1,2,4,8, ... bytů (např. pro 1 MB, potřebujeme 21 takových seznamů).



Zřetězené seznamy (6)

- **Výhody**

- alokace paměti je stejně rychlá jako u quick fit algoritmu,
- slučování sousedních děr při uvolnění paměti je rychlejší než u quick fit algoritmu (nemusíme procházet všechny seznamy)

- **Použití**

- např. pro alokaci paměti v jádře Linuxu