



PB153

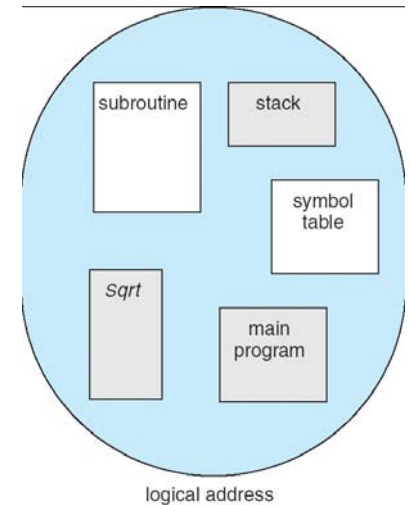
Operační systémy a jejich rozhraní

Správa paměti

Principy, základy

- Pro běh procesu je nutné, aby program, který je vykonáván byl umístěn v operační paměti (hlavní paměti)
- Z programu se stává proces (aktivní entita schopná spuštění na CPU) provedením celé řady kroků
 - naplnění tabulek, umístění do operační paměti
 - vázání adres instrukcí a dat na adresy operační paměti

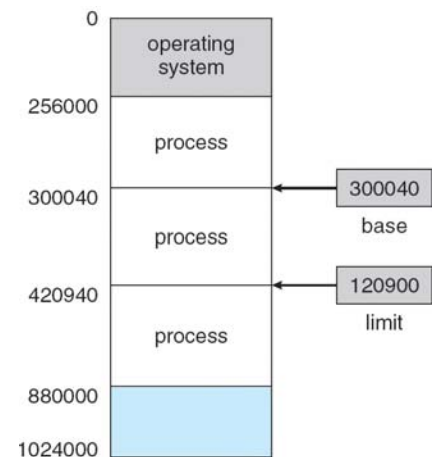
Správa paměti



- Program je složen z částí se vzájemně odlišnými vlastnostmi
 - moduly s instrukcemi jsou označovány „jen ke spuštění“ (execute-only).
 - datové moduly jsou buďto „read-only“ nebo „read/write“
 - některé moduly jsou „soukromé“ (private), jiné jsou veřejné „public“
- Více procesů může sdílet společnou část paměti, aniž by se tím porušovala ochrana paměti
 - neboť sdílení jedné datové struktury je lepší řešení než udržování konzistence násobných kopií vlastněných jednotlivými procesy

Vázání adres – možnosti

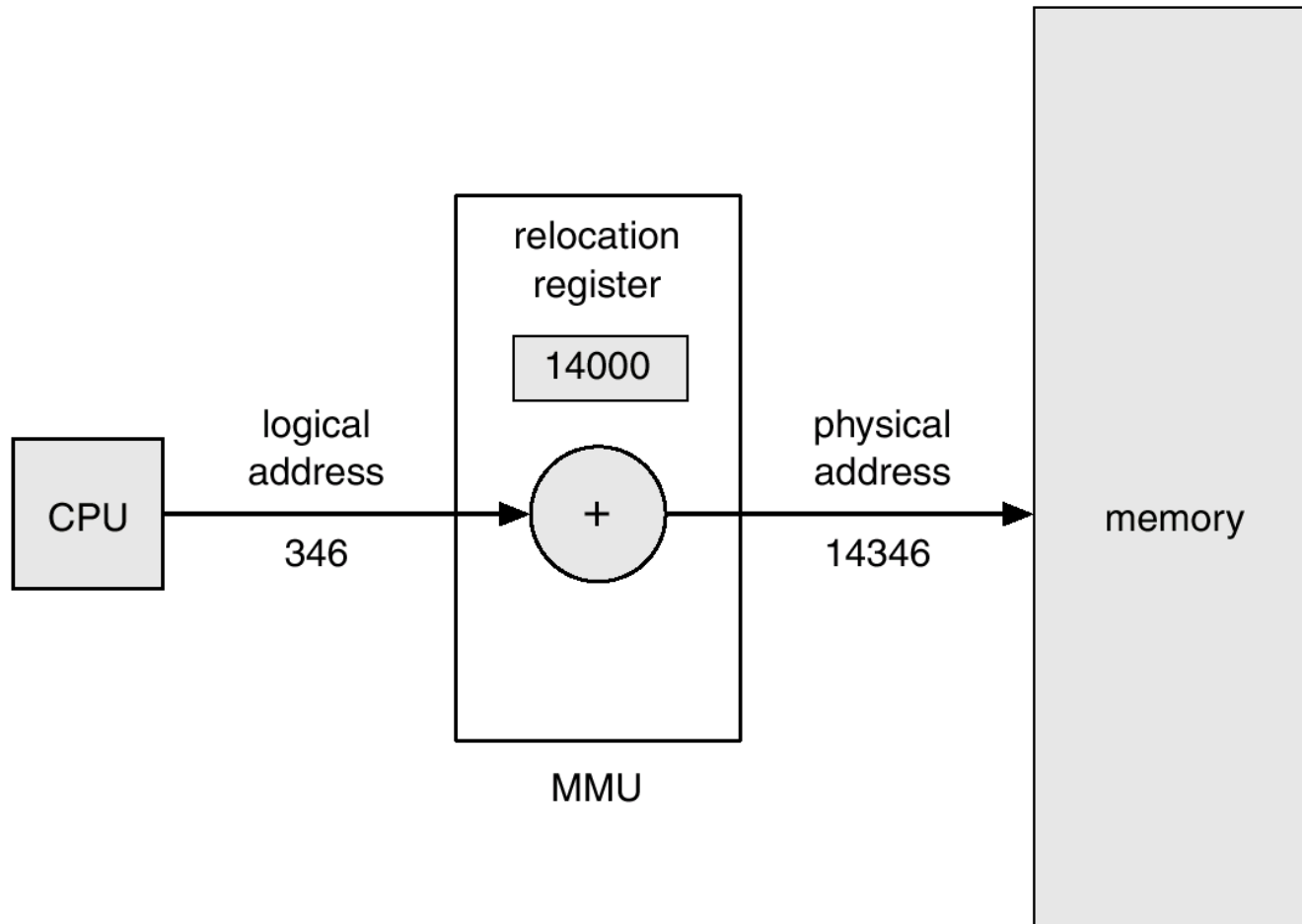
- Při kompilaci
 - umístění v paměti je známé a priori
 - lze generovat absolutní kód
 - při změně umístění se musí program znovu přeložit
- Při zavádění
 - umístění v paměti není známé v době kompilace
 - generuje se přemístitelný kód (relocatable code).
- Za běhu
 - jestliže proces může měnit svoji polohu během provádění, vázání se zpožďuje na dobu běhu
 - musí být dostupná hardwarová podpora
 - bazové registry, mezní registry, ...



Memory-Management Unit

- Hardwarový modul převádějící logické adresy na fyzické adresy
- Uživatelský program pracuje s logickými adresami, uživatelský program nevidí fyzické adresy
- Připočítává se obsah „relokačního registru“ k adresám generovaným uživatelským procesem v okamžiku, kdy je předávána jako ukazatel do operační paměti

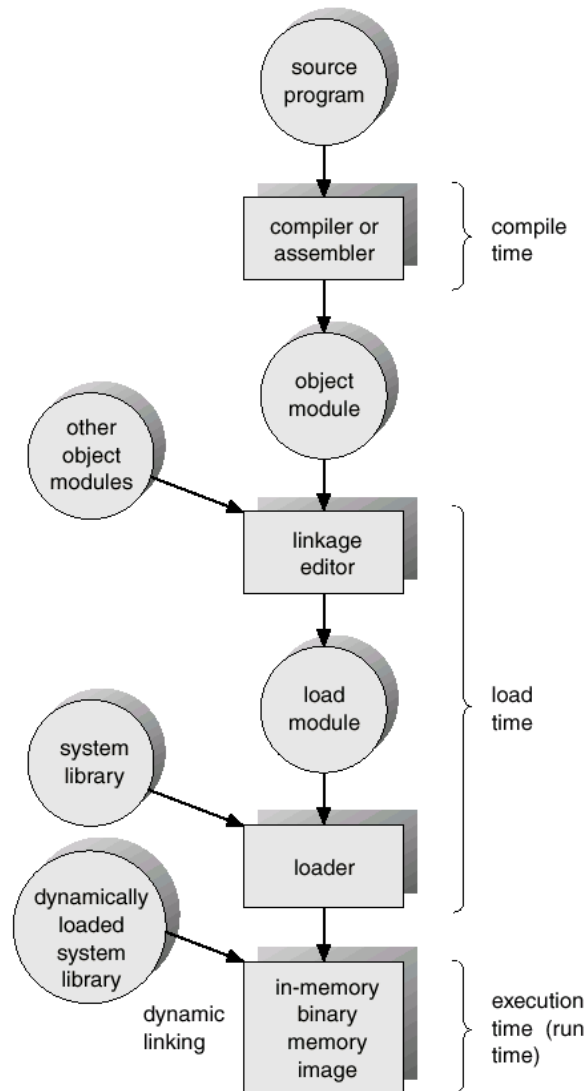
Relokační registr



Adresový prostor

- Logický adresový prostor (LAP), fyzický adresový prostor (FAP)
 - LAP – (logická adresa, virtuální adresa) dána adresou ve strojovém jazyku, generuje CPU
 - FAP – (fyzická) adresa akceptovaná operační paměť
- Logické a fyzické adresové prostory se shodují v době kompilace a v době zavádění
- Logické a fyzické adresové prostory mohou být rozdílné při vázání v době běhu

Proces vytváření programu

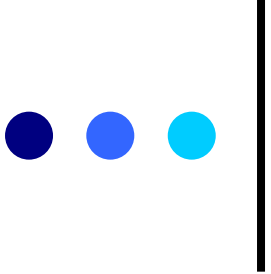


● ● ● | Dynamické zavádění

- Dynamic loading
- Programový kód se nezavádí dokud není zavolán (spuštěn)
- Dosahuje se lepšího využití paměti
 - nepoužívané části kódu se nikdy nezavádí
- Užitečná technika v případech, kdy se musí velkými programovými moduly řešit zřídka se vyskytující alternativy
- Program nevyžaduje žádnou speciální podporu od operačního systému

● ● ● | Dynamické vázání

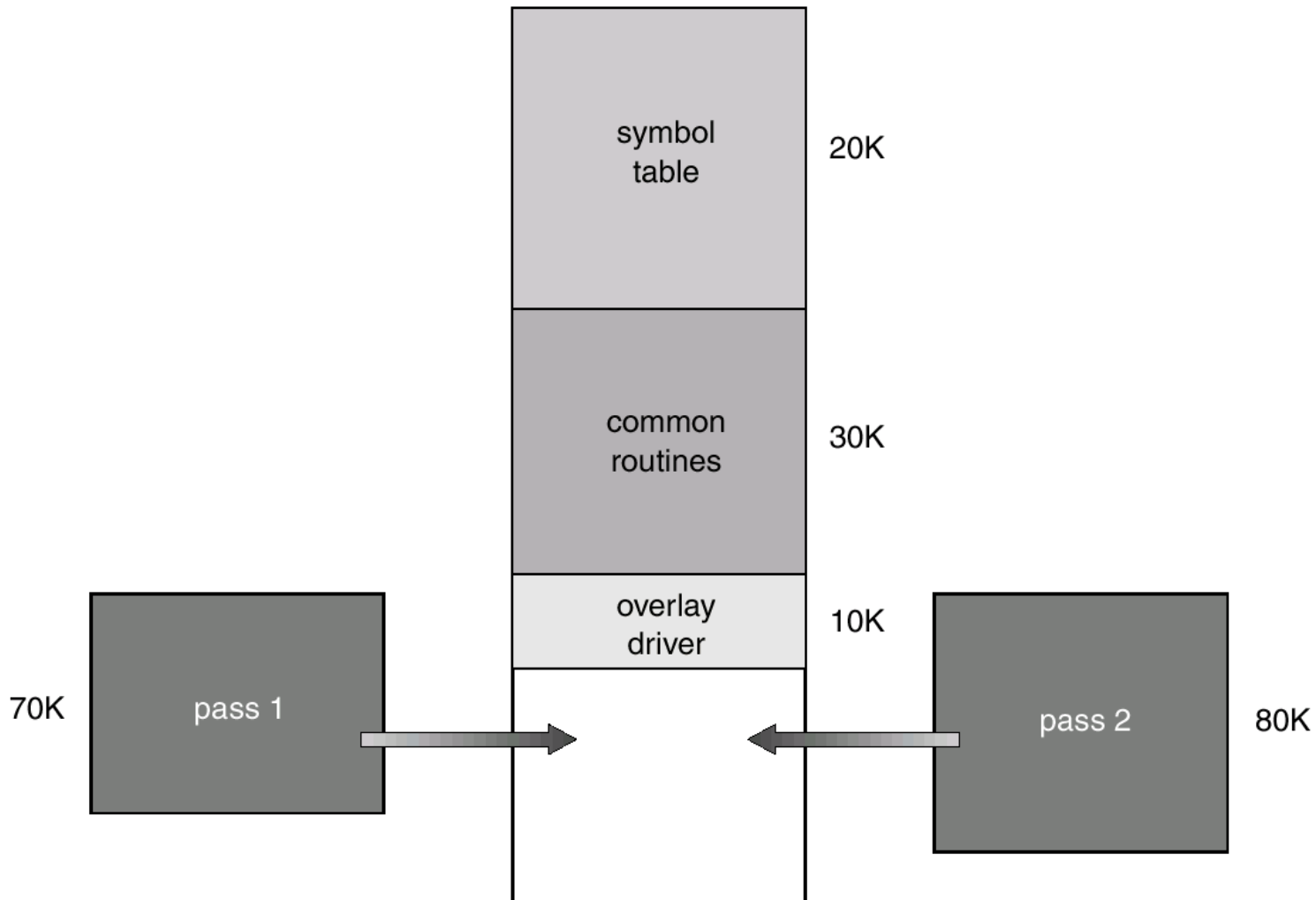
- Dynamic linking
- Vázání je odkládáno na dobu běhu
- Pro umístění příslušných knihovních programů rezidentních v operační paměti se používá kód malého rozsahu – tzv. stub
- Při zavolání stub nahradí sám sebe adresou skutečné funkce a předá jí řízení
- OS musí kontrolovat, zda funkce je mapována do paměti procesu
- Je to technika vhodná zvláště pro knihovní funkce



Překryvy, Overlays

- V operační paměti se uchovávají pouze ty instrukce a data, která jsou potřeba po celou dobu běhu
- Technika, která je nutná v případech, kdy je přidělený prostor paměti menší než je souhrn potřeb procesu
- Vyvolávání překryvů je implementované programátorem, od OS se nepožaduje žádná speciální podpora
- Návrh překryvové struktury ze strany programátora je značně složitý

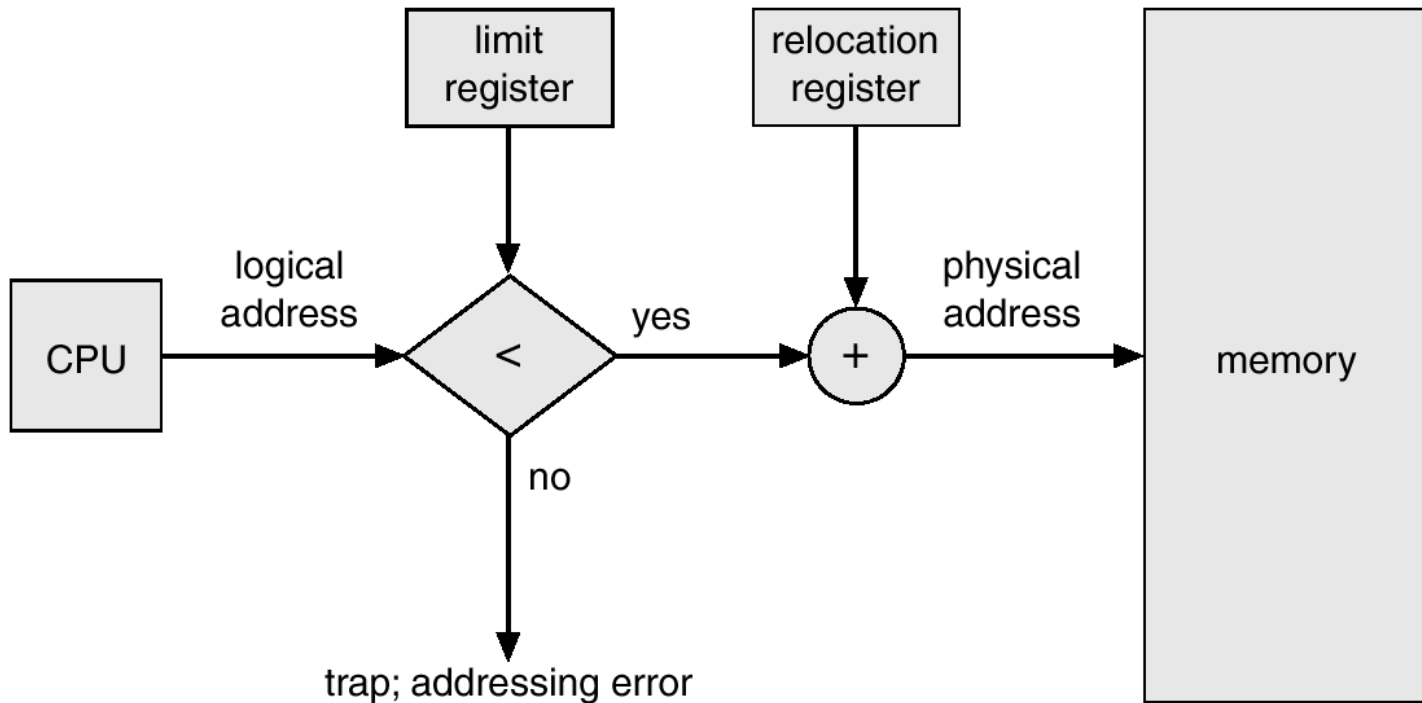
Overlay: příklad



Souvislé oblasti

- Operační paměť se dělí do dvou sekcí
 - rezidentní OS, obvykle na počátku FAP s tabulkou ovladačů přerušení
 - uživatelské procesy
- Přidělování jedné souvislé části paměti
 - pro ochranu procesů uživatelů mezi sebou a OS lze použít schéma s relokačním registrem
 - relokační registr: hodnota nejmenší fyzické adresy paměti procesu
 - mezní registr: rozpětí logických adres, logická adresa musí být menší nebo rovna meznímu registru

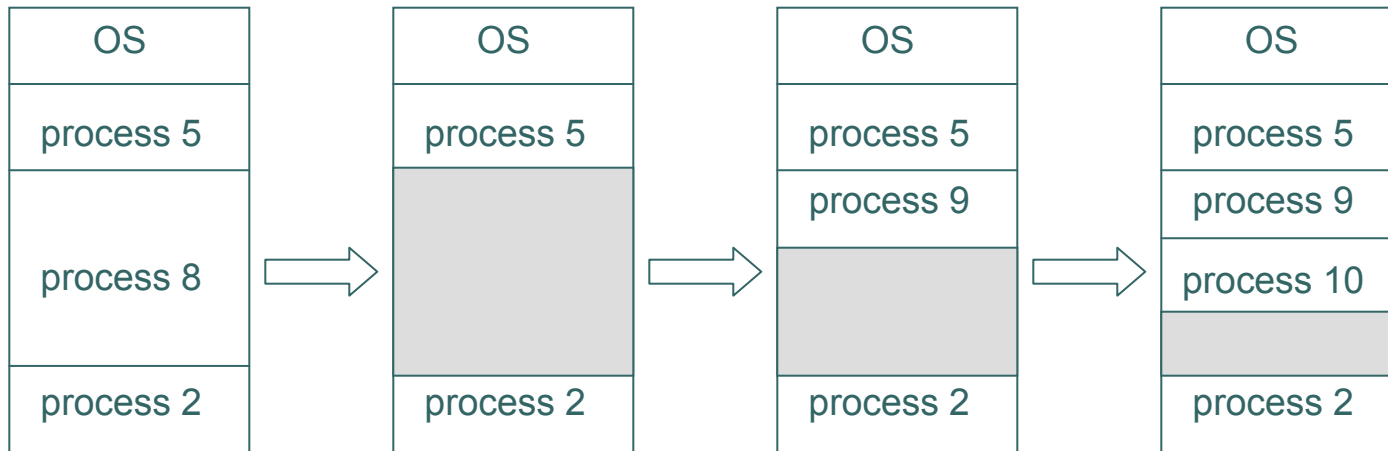
HW podpora



Souvislé oblasti

○ Přidělování několika částí paměti

- *díra* – blok dostupné paměti
 - Bloky jsou roztroušeny po FAP
- evidenci o přidělených a volných sekcích udržuje OS





Přidělování paměti

- Kterou oblast délky n přidělit, když volná paměť je rozmístěna ve více souvislých nesousedních sekcích?
 - First-fit:
 - přiděluje se první dostatečně dlouhá volná oblast resp. její počátek
 - Best-fit:
 - přiděluje se nejmenší dostatečně dlouhá volná oblast resp. její počátek
 - generují se velmi malé (nejmenší) možné volné díry
 - Worst-fit:
 - přiděluje se největší dostatečně dlouhá volná oblast resp. její počátek
 - generují se největší možné volné díry
- Z hlediska rychlosti a kvality využití paměti jsou First-fit a Best-fit jsou lepší techniky než technika Worst-fit



Problém fragmentace

- Vnější fragmentace
 - souhrn volné paměti je dostatečný, ale ne v dostatečné souvislé oblasti
- vnitřní fragmentace
 - přidělená oblast paměti je větší než požadovaná velikost, tj. část přidělené paměti je nevyužitá
- Snižování vnější fragmentace setřásáním
 - přesouvají se obsahy paměti s cílem vytvořit (jeden) velký volný blok
 - použitelné jen když je možná dynamická relokační (viz MMU)
 - provádí se v době běhu
 - problém I/O
 - s vyrovnávacími paměťmi plněnými z periférií autonomně nelze hýbat – umisťují se proto do prostoru OS



Stránkování

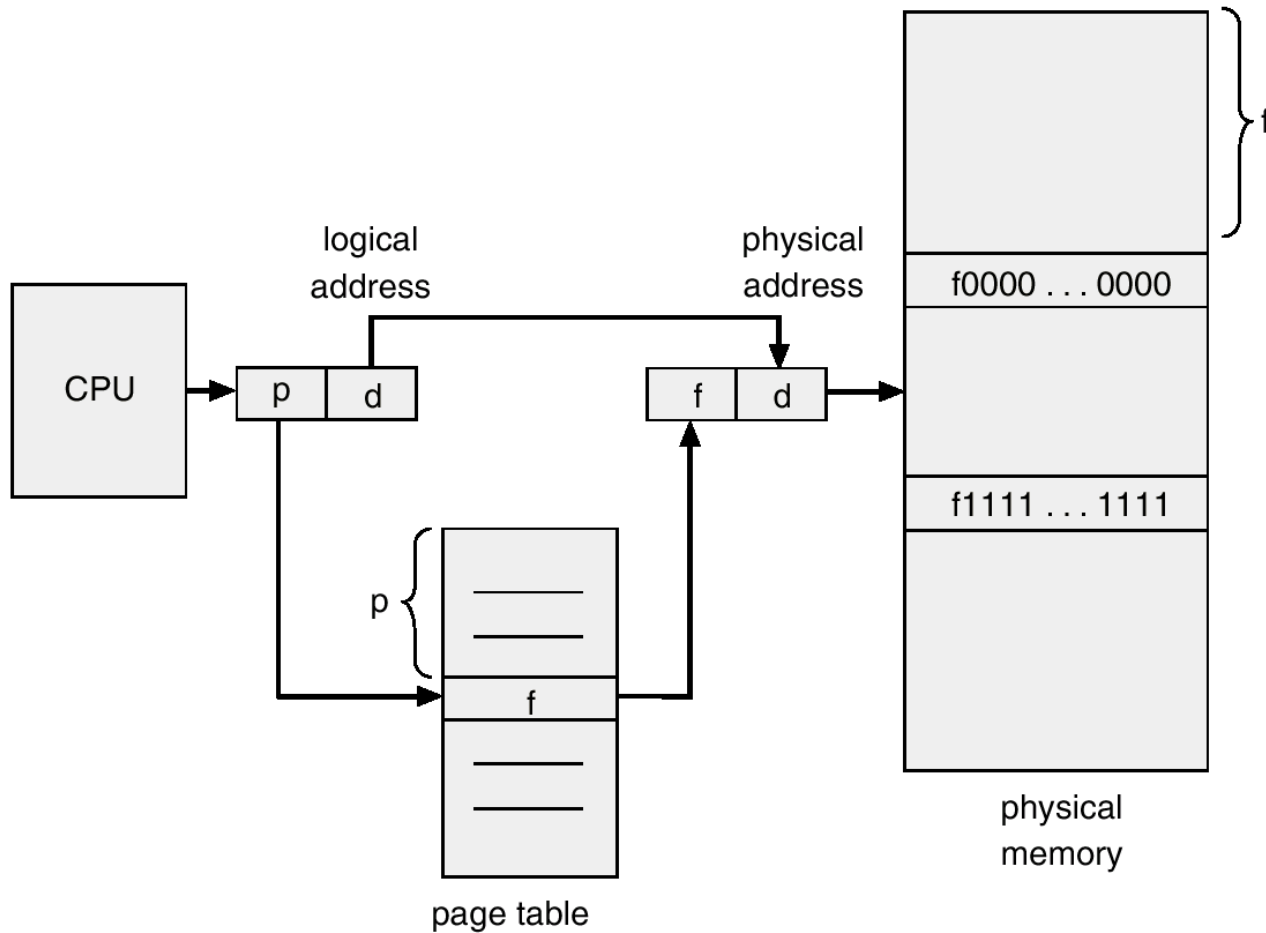
- LAP procesu nemusí být jedinou souvislou sekcí FAP, LAP se zobrazuje do (po částech volných) sekcí FAP
- FAP se dělí na sekce zvané rámce (frames)
 - pevná délka, délka v násobcích mocnin 2 (obvykle mezi 512 až 8192 bajty)
- LAP se dělí na sekce zvané stránky (pages)
 - pevná délka, shodná s délkou rámců
- Udržujeme seznam volných rámců
- Program délky n stránek se umístí (zavede) do n rámců
- Překlad logická adresa \rightarrow fyzická adresa – pomocí překladové tabulky nastavované OS a interpretované MMU
- Vzniká vnitřní fragmentace, neboť paměť je procesu přidělována v násobcích velikosti rámce

Dynamický překlad adres

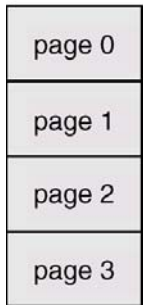
- Logická adresa (generovaná CPU) se dělí
 - číslo stránky, p
 - index do tabulky stránek
 - index bazové adresy rámce přiděleného stránce, které patří logická adresa
 - offset ve stránce, d
 - přičítáme k začátku stránky/rámce



Příklad stránkování



Příklad stránkování (2)

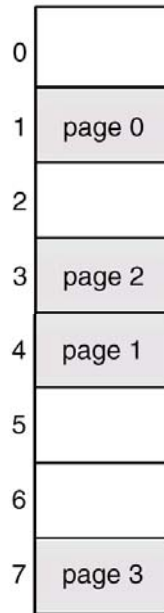


logical memory

0	1
1	4
2	3
3	7

page table

frame number



physical memory

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

logical memory

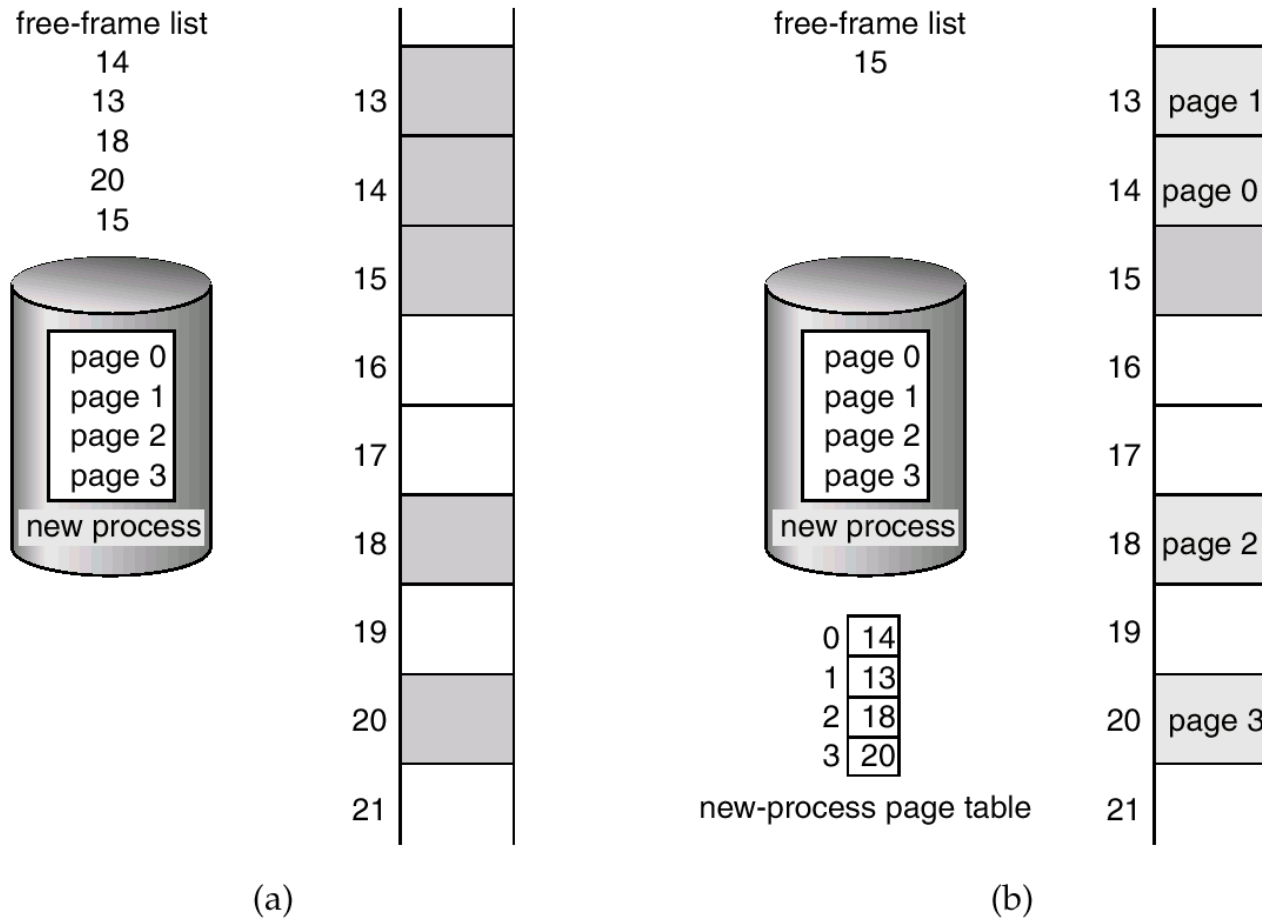
0	5
1	6
2	1
3	2

page table

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

physical memory

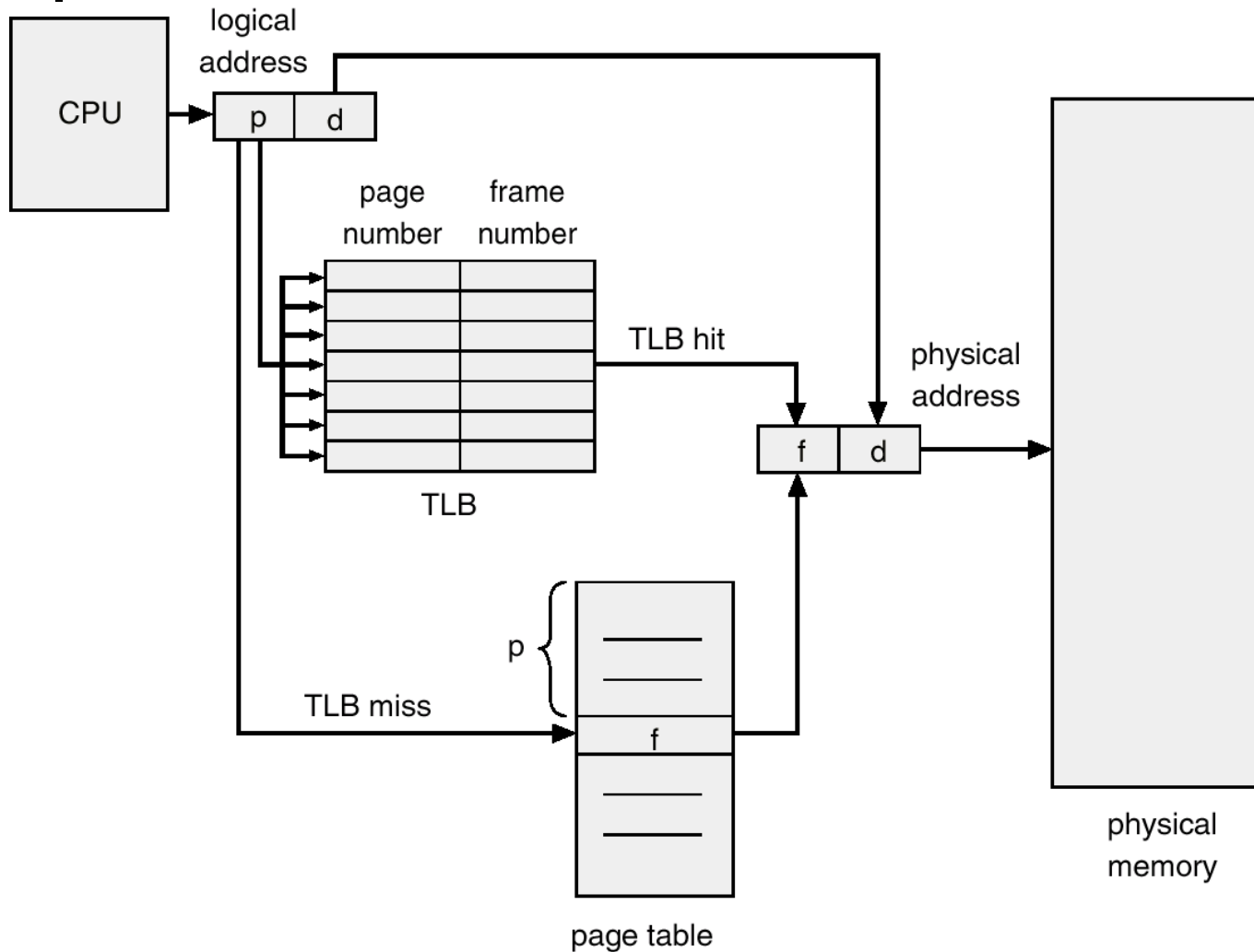
Příklad stránkování (3)



Tabulka stránek

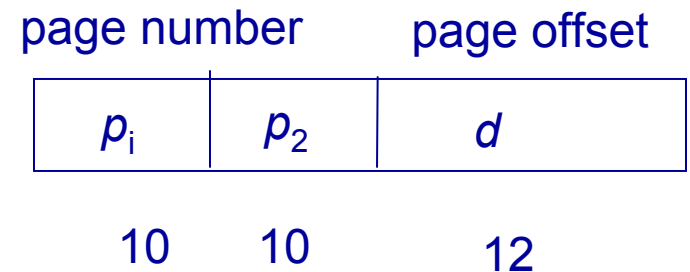
- Je uložena v operační paměti
- Její počátek a konec je odkazován registrem Page-table base register (PTBR), Page-table length register (PTLR)
- Zpřístupnění údaje / instrukce v operační paměti vyžaduje dva přístupy do operační paměti
 - jednou do tabulky stránek
 - jednou pro údaj/instrukci
- Problém zhoršení efektivity dvojitým přístupem lze řešit speciální rychlou hardwarovou cache pamětí
 - asociativní paměť
 - translation look-aside buffers (TLBs)

Stránkování s TLB

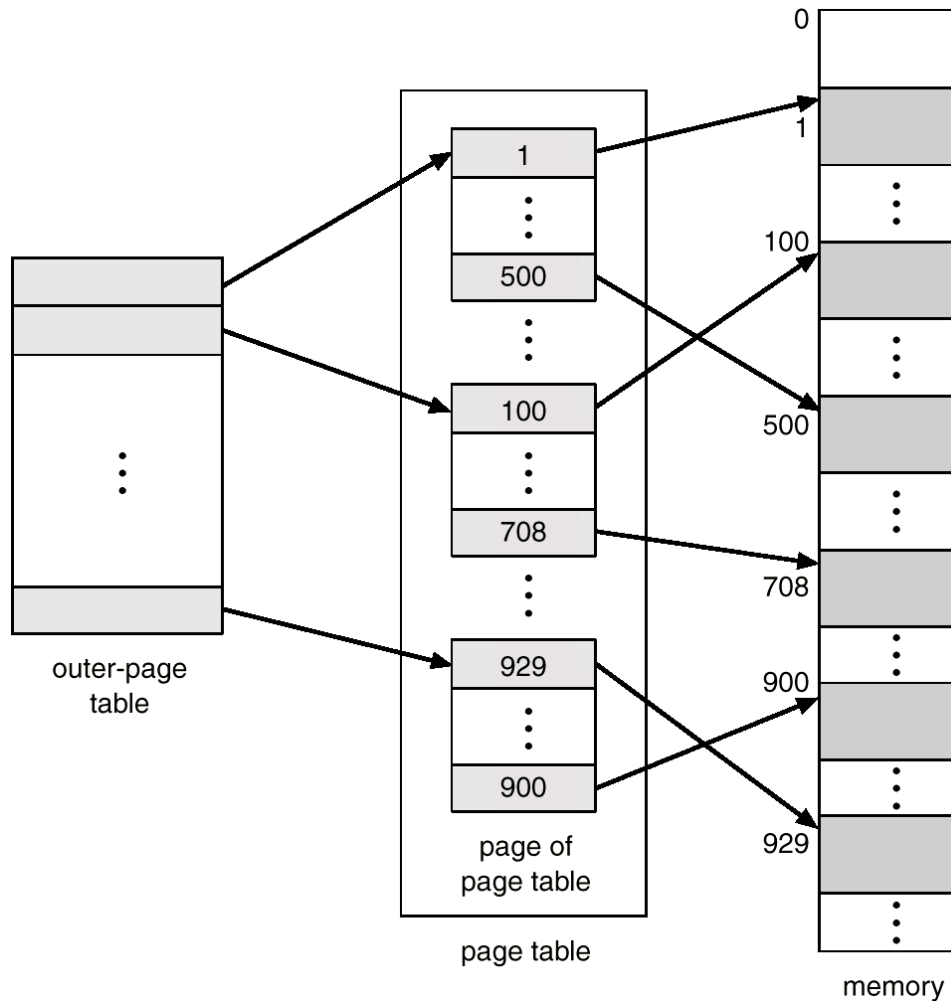


Dvouúrovňová tabulka stránek

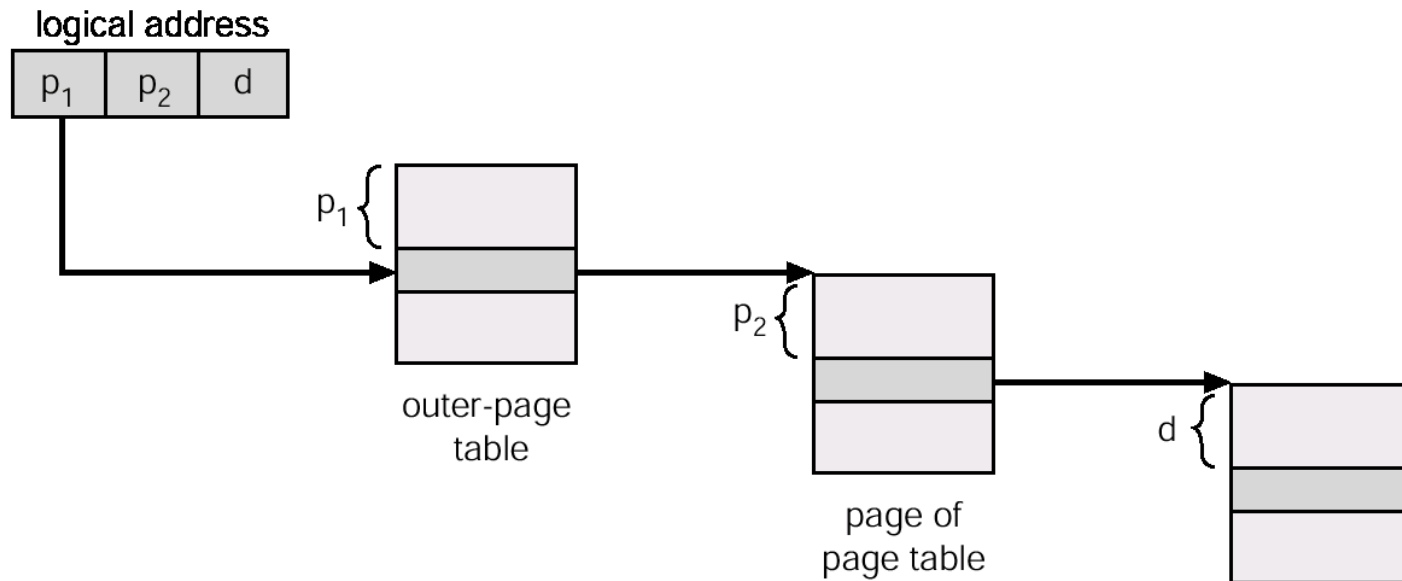
- 32-bitový procesor s 4KB stránkou
- PT (tabulka stránek) je stránkovaná
- Logická adresa
 - číslo stránky: 20 bitů
 - adresa ve stránce: 12 bitů
- Číslo stránky se dále dělí
 - číslo stránky 10-bitů
 - adresa v tabulce stránek 10-bitů



Dvouúrovňová tabulka stránek



Tvorba adresy

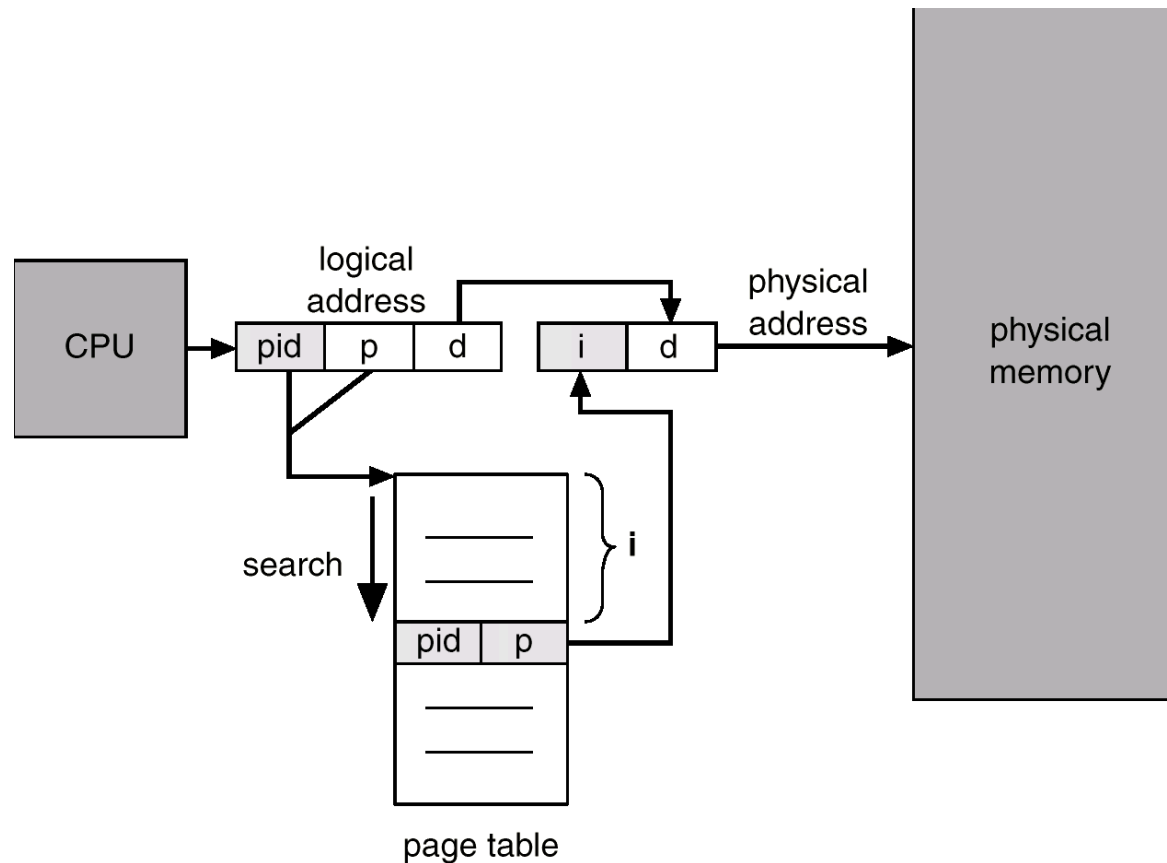


Invertovaná tabulka stránek

- 1 položka v PT pro každý rámeček paměti
- Obsahuje
 - logickou (virtuální) adresu stránky uchovávané v odpovídajícím rámečku
 - informaci o procesu, který stránku vlastní
- Snižuje se velikost paměti potřebné pro uchování PT
- Zvyšuje se doba přístupu do PT, indexový přístup je nahrazen prohledáváním
 - lze zvýšit efektivnost využitím hašování
 - perfektní hašování – 1 přístup k PT
 - jinak několik přístupů k PT

Invertovaná tabulka: příklad

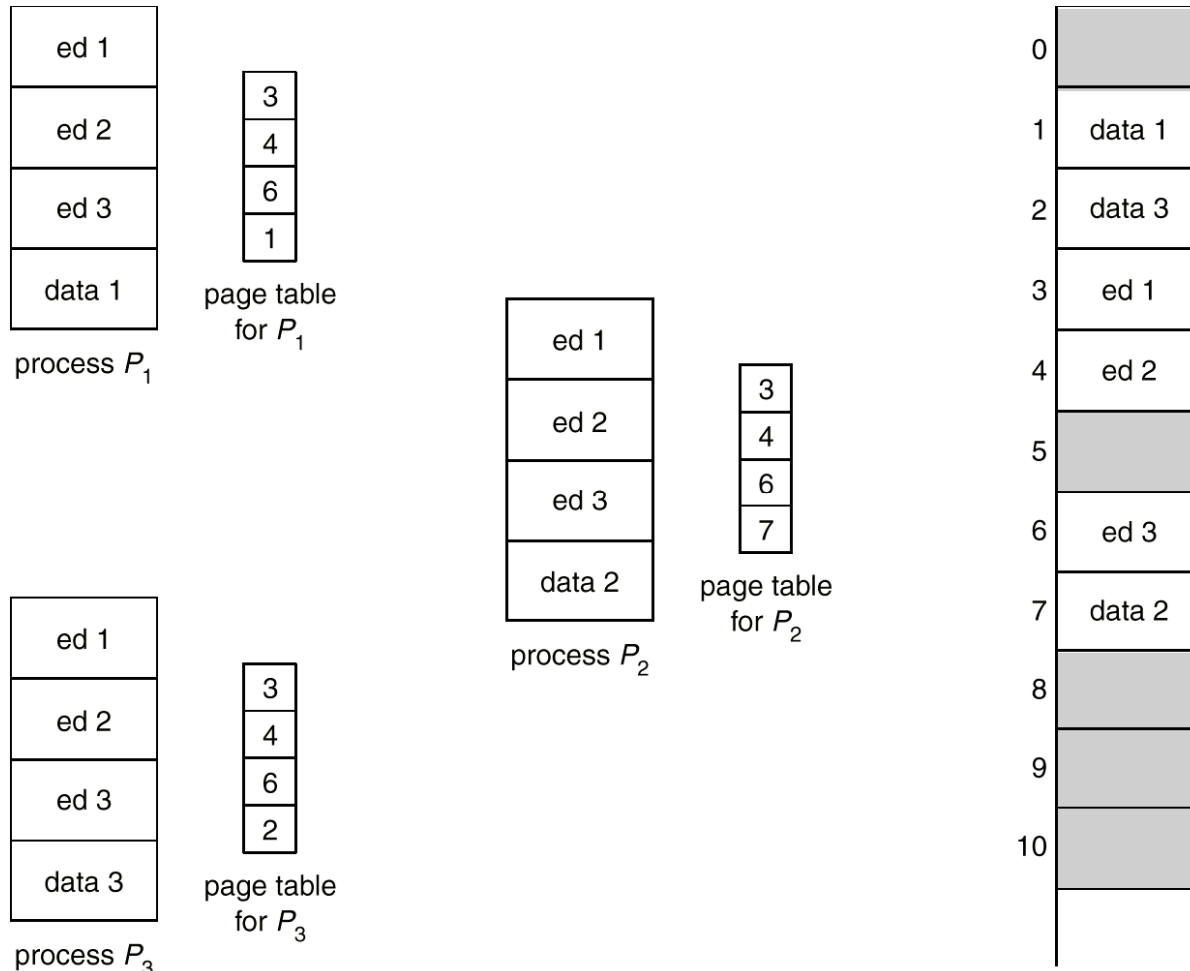
- Např. AS400 (IBM), UltraSPARC, PowerPC



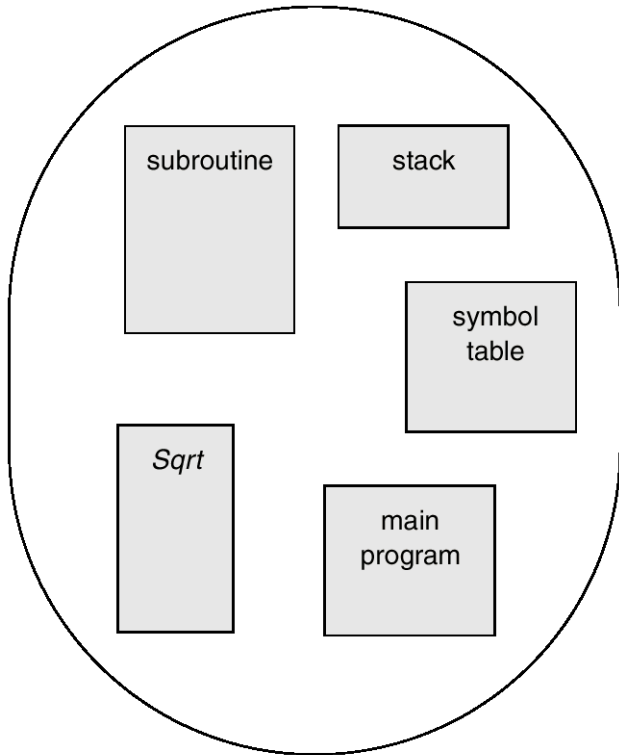
Sdílení stránek

- Sdílený kód je umístěn v paměti pouze jednou
 - kód je reentrantní a „read-only“
 - musí se nacházet na stejné logické adrese ve všech procesech
- Privátní data
 - každý proces udržuje svoji privátní kopii dat (a nesdíleného kódu)

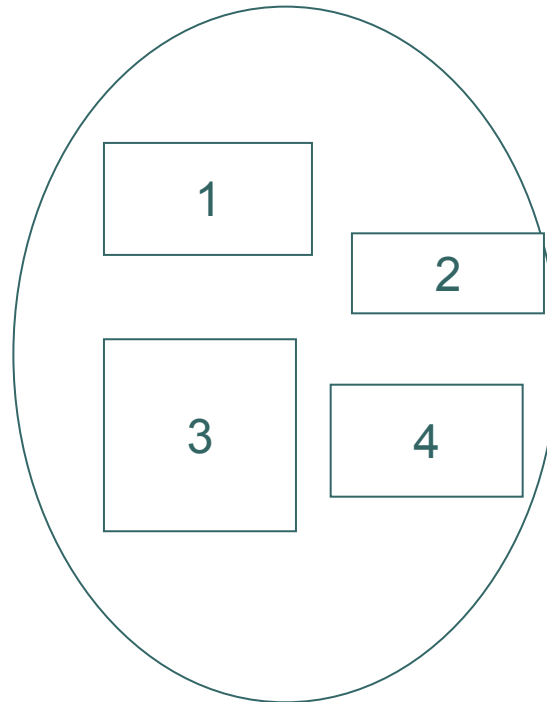
Sdílení stránek: příklad



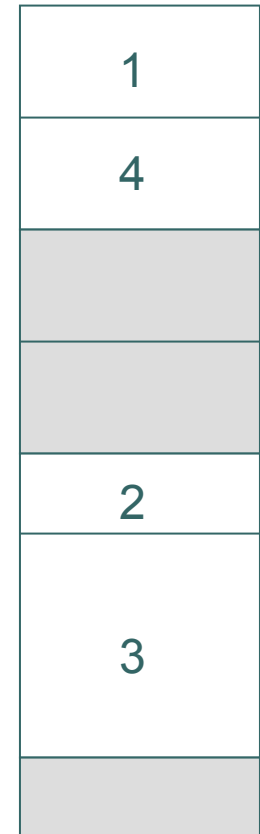
Segmentování



logical address space



user space

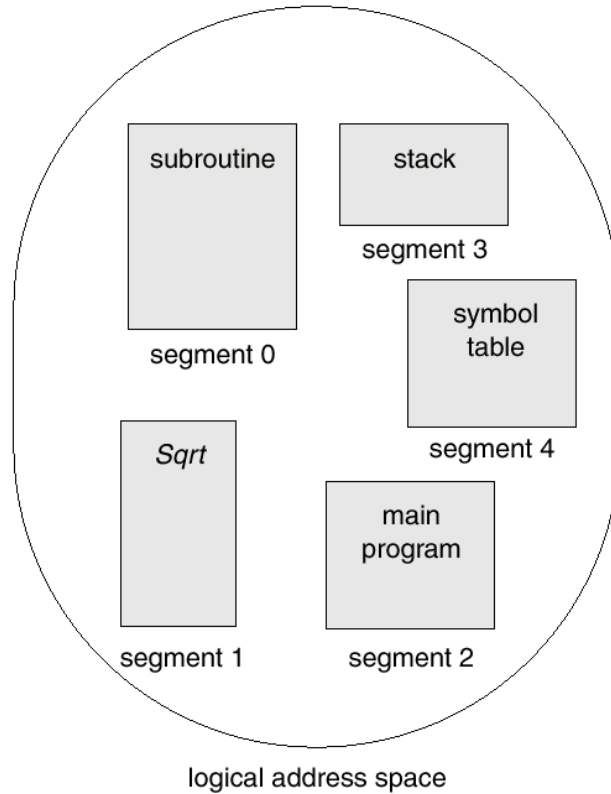


physical memory space

Segmentování

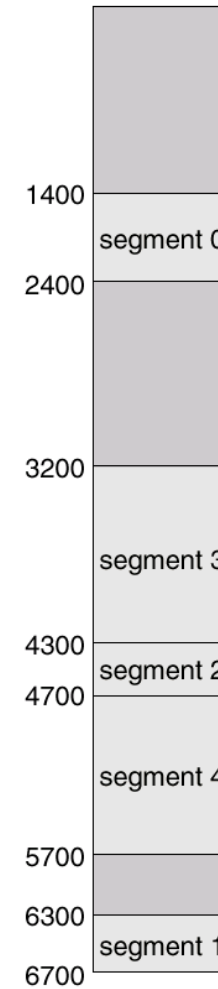
- Logická adresa je dvojice (segment s , offset d)
- *Tabulka segmentů, Segment table, ST*
 - *base* – počáteční adresa umístění segmentu ve FAP
 - *limit* – délka segmentu
- *Segment-table base register (STBR)*
 - odkaz na umístění ST v paměti
- *Segment-table length register (STLR)*
 - počet segmentů, s je legální když $s < STLR$
- Relokace – dynamická, pomocí ST

Příklad segmentace

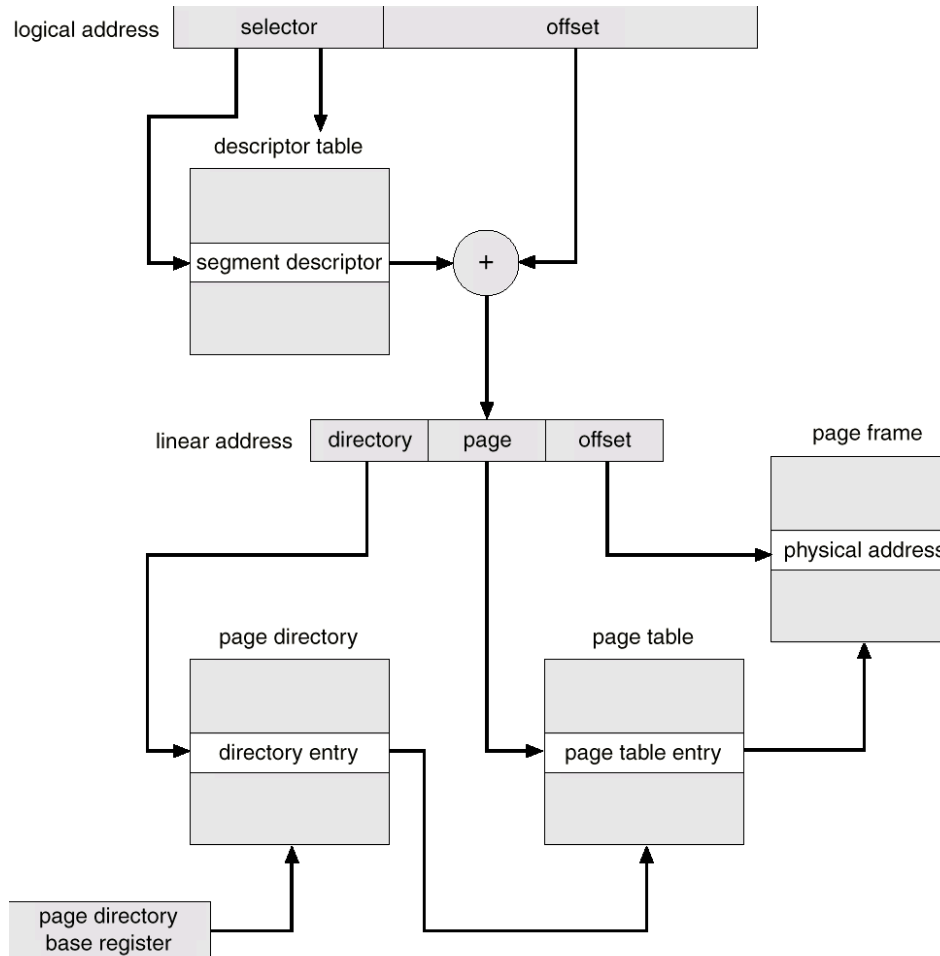


	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

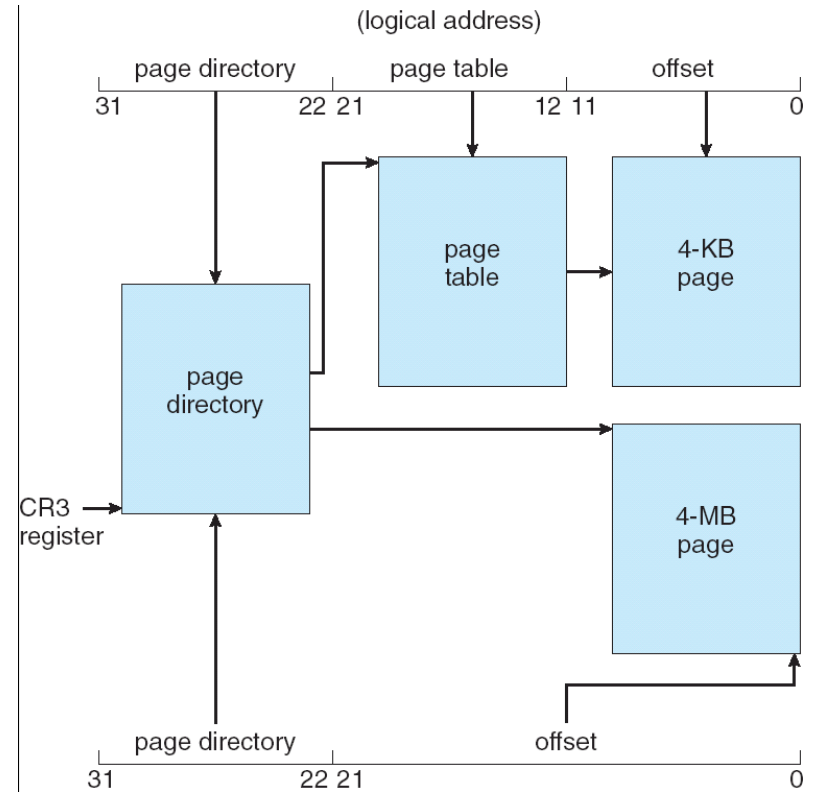
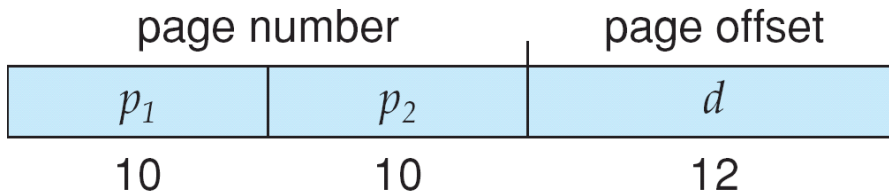
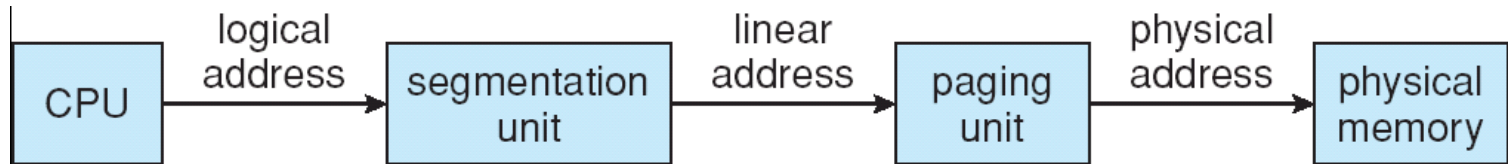
segment table



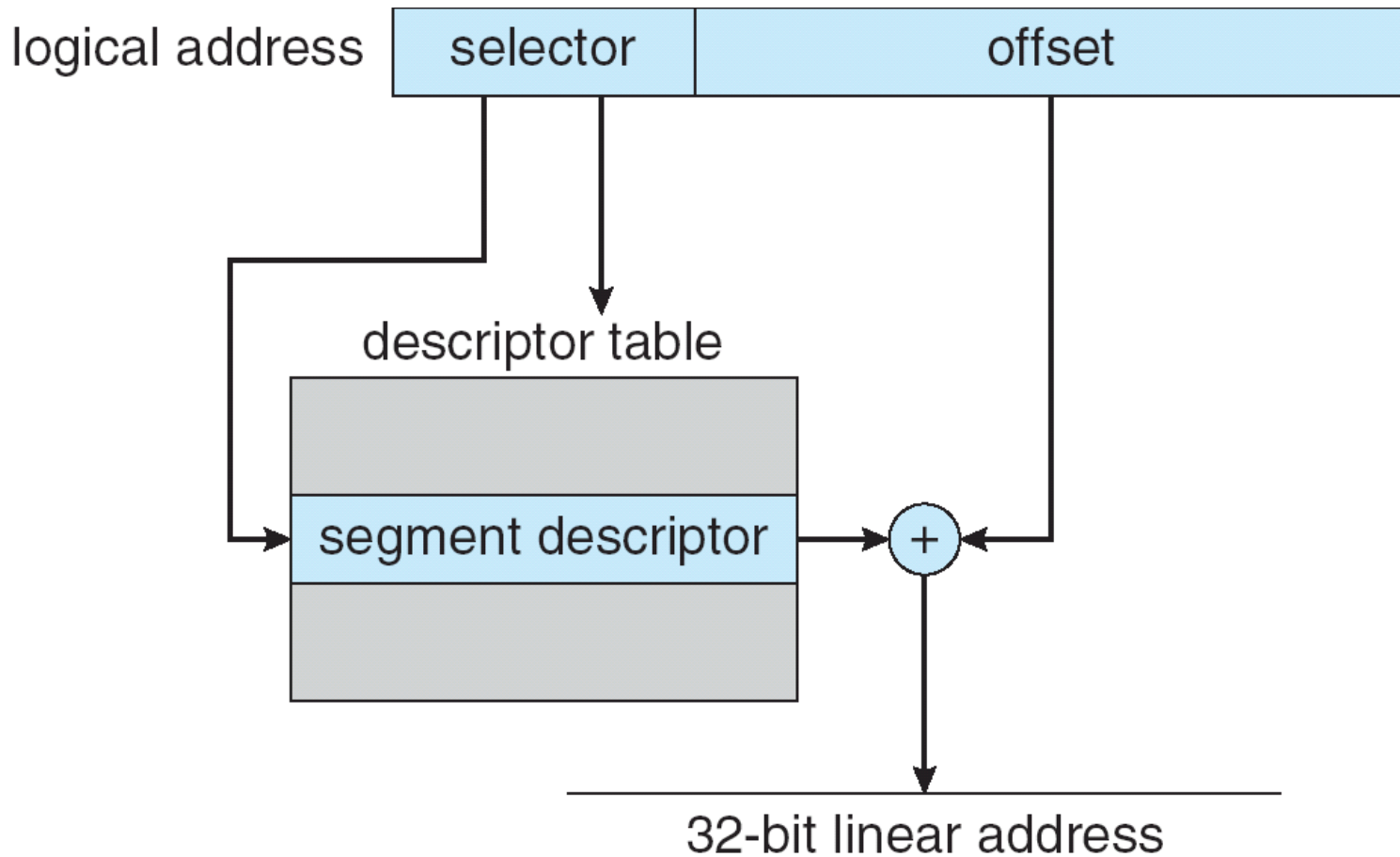
Stránkování a segmentování (Intel 386)



Příklad: Intel Pentium



● ● ● | Příklad: Intel Pentium



Příklad: Linux

