SPRÁVA PAMÄTE

I časť

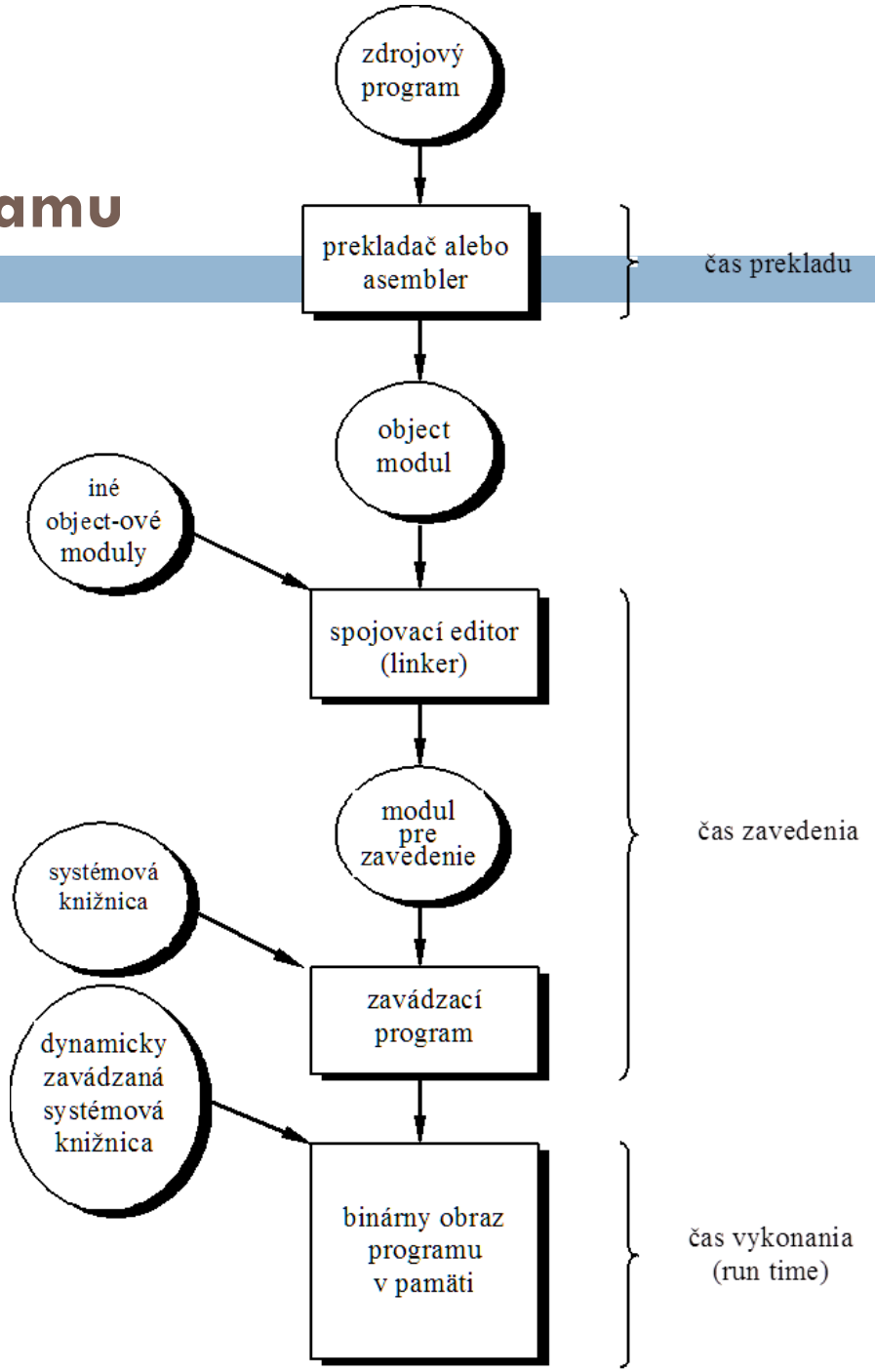
Obsah prednášky

- **Úvod**
- **Logický a fyzický adresný priestor**
- **Swapovanie**
- **Súvislé prideľovanie pamäte**
- **Stránkovanie**
- **Segmentácia**

Úvod

- Kompilácia programu
  - preklad symbolických názvov do logických adries
- Spracovanie inštrukcie - dekódovanie, odkazy na pamäť
- Pripojenie fyzických adries (obr.)
  - **počas prekladu**
    - súbory typu *.com* MS DOSu, zmena ->opätovný preklad, zriedkavé použitie, niekedy pre komponenty OS
  - **počas zavadzania**
    - relokovateľný kód, zmena -> opätovný zavedenie
  - **počas vykonania**

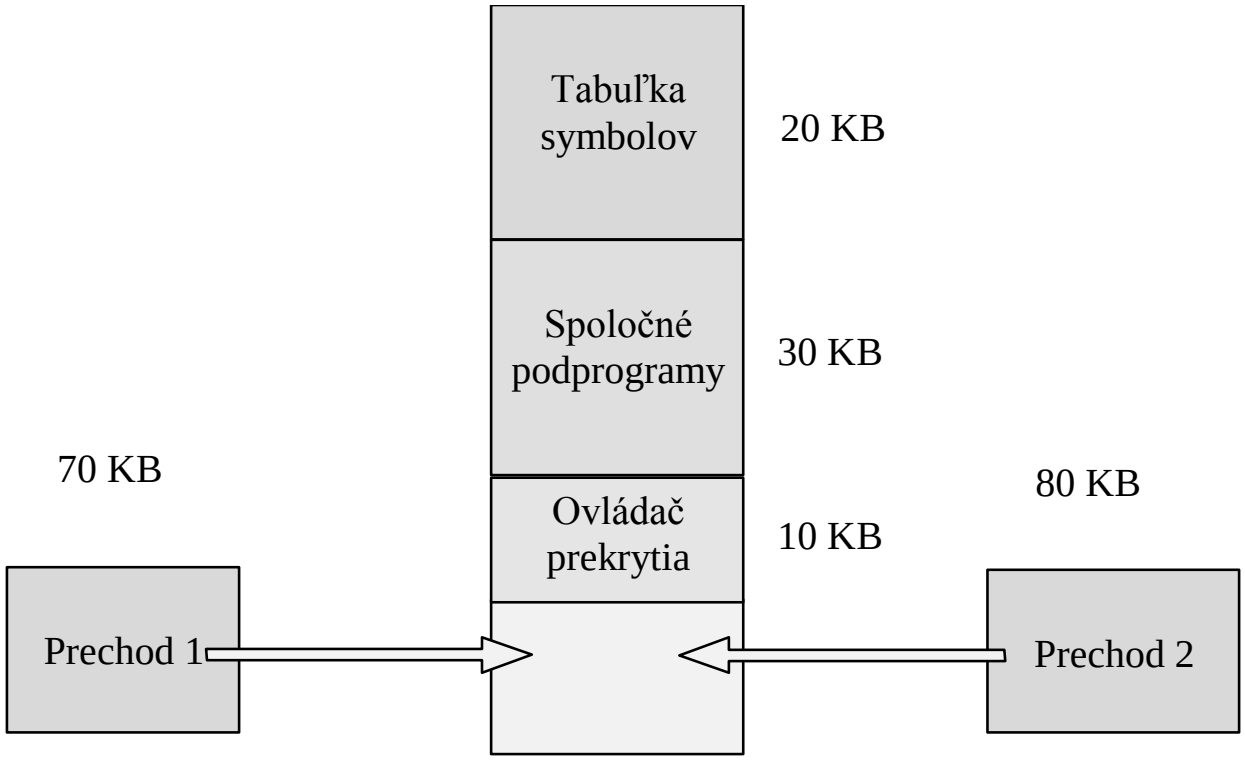
Kroky spracovania používateľského programu



Spolupráca s používateľom

- Dynamické zavádzanie
  - podprogramy sa nezavádzajú do pamäte, kým nie sú volané;
  - nevyžaduje špeciálnu podporu zo strany operačného systému

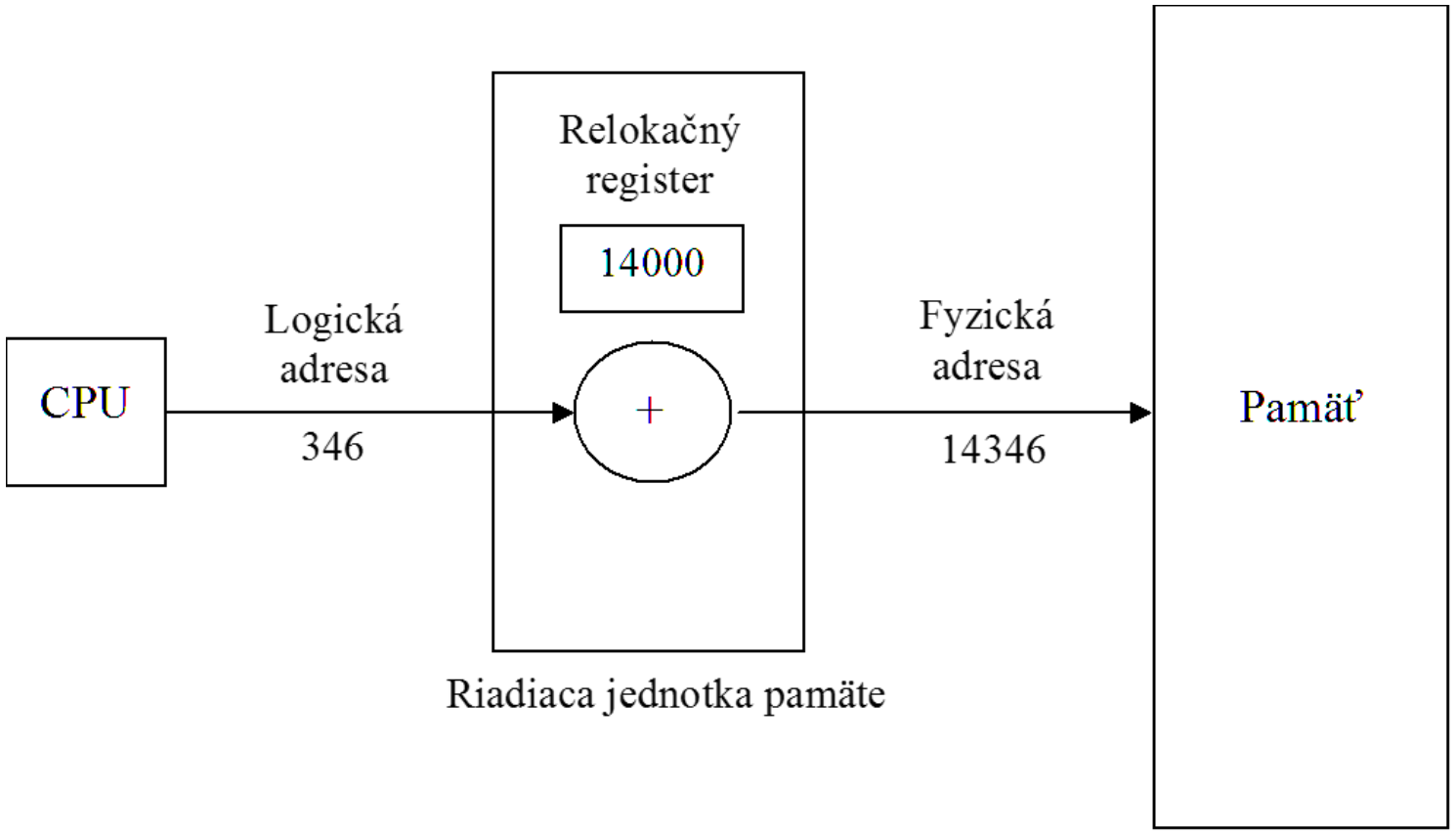
Prekrytia v dvojprechodovom asembleri



Logický a fyzický adresný priestor

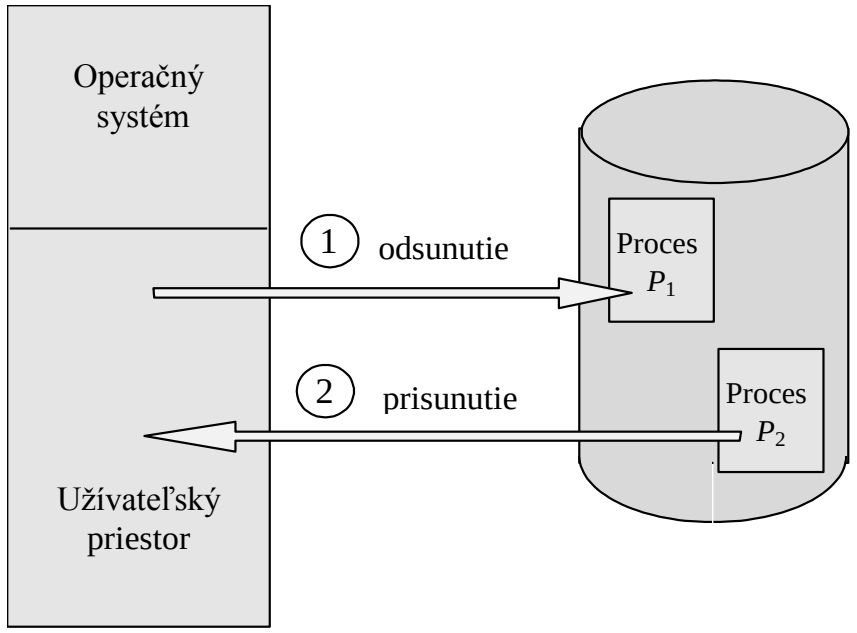
- **Logické adresy**
  - adresy generované procesorom
- **Fyzické adresy**
  - adresy, ktoré používa MMU (tie ktoré sa zavádzajú do registra pamäťových adries)
- **LAP**
  - množina logických adries, ktoré sú generované programom
- **FAP**
  - množina fyzických adries, ktoré odpovedajú logickým adresám
- **Memory Management Unit** (MMU) –
  - mapuje logické adresy na fyzické počas behu programu

Dynamická relokácia využívajúca relokačný register



Swapovanie

- ***Dočasne odsunutie procesu z pamäte na disk a neskôr opätovne vrátenie do pamäte***
  - Dispečer po spustení naplánovaného procesu kontroluje, či ďalší proces z frontu je v pamäti. Ak nie je a nie je voľné miesto v pamäti, odsúva (swap out) niektorý z procesov v pamäti a prisúva (swap in) požadovaný proces



Swapovanie pokračovanie

- Príklad:

  Čas pre swapovanie

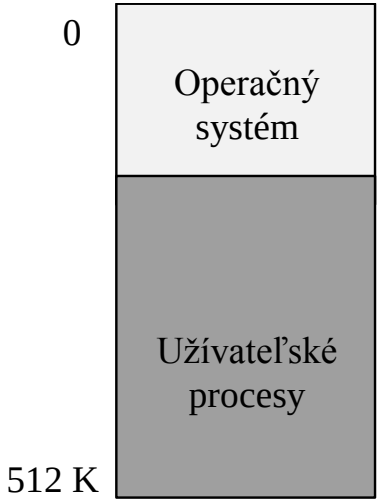
  proces má veľkosť 100KB, disk s rýchlosťou 1MB/s.
  Prenos zaberie:

  100/1000 KB za sekundu = 1/10 s = **100 ms**

  reakčný čas - 8 ms, celkový čas - 108 ms
- Čas pre swapovanie je úmerný veľkosti presúvaných procesov
- Swapovanie a odštartované V/V operácie - problém
- Sú možné dve riešenia tohto problému:
  - **nikdy neodsúvať proces s nedokončenými V/V operáciami.**
  - vykonávať **V/V operácie len cez bufre operačného systému.**
- Väčšina systémov používa nejaký variant swapovania

Súvislé prideľovanie pamäte

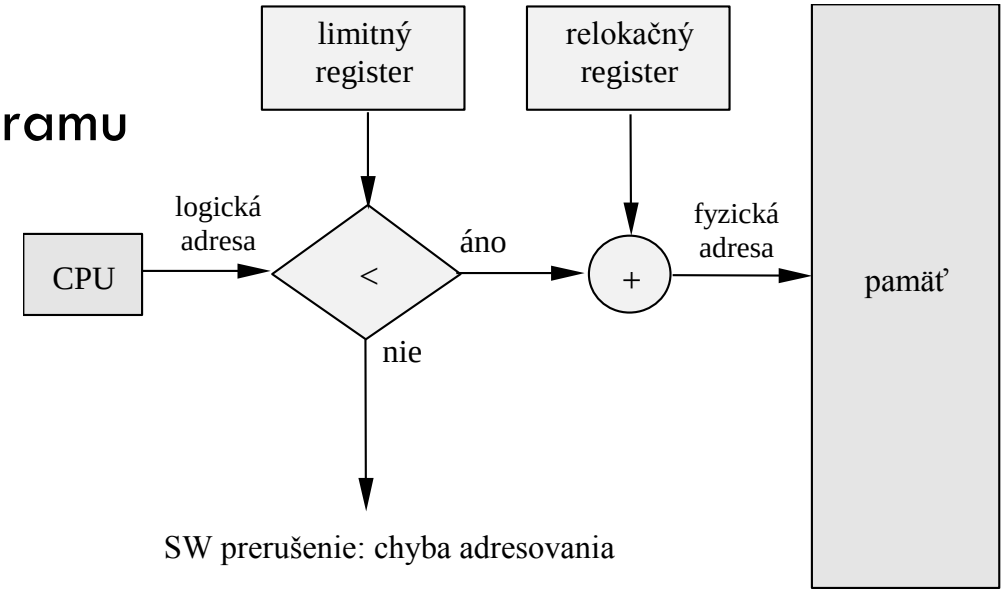
- Prideľovanie jedného úseku



  - je najjednoduchšou technikou správy pamäte, všetkým procesom prideľuje ten istý úsek,
  - typická pre monoužívateľské systémy bez paralelného spracovania (CP/M, MS-DOS),
  - nepoužíva sa už.

Súvislé prideľovanie pamäte pokračovanie

- Prideľovanie **viacerých súvislých úsekov s pevnou dĺžkou –** multiprogramovanie
  - prideľuje sa pri zavadzaní programu
  - ochrana
    - relokačný register
    - limitný register

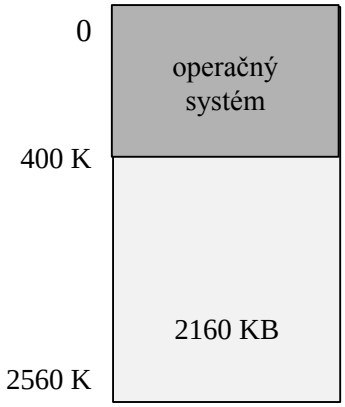


  - technika, použitá v systéme IBM OS/360, známa pod názvom MFT (Multiprogramming with a Fixed number of Tasks).

Súvislé prideľovanie pamäte pokračovanie

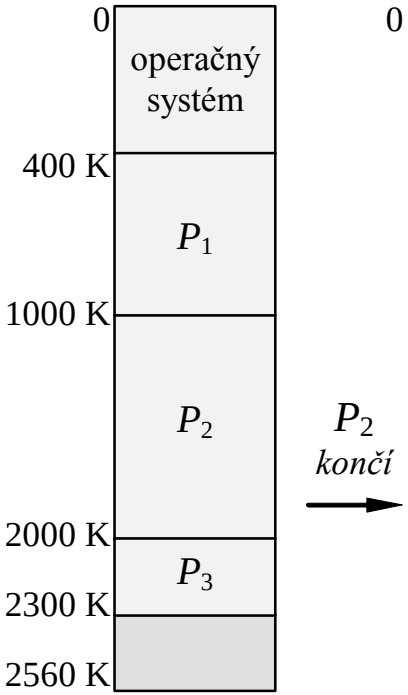
- Prideľovanie súvislých úsekov s premenlivou dĺžkou
  - rozmery úsekov sa menia dynamicky s veľkosťou vznikajúcich procesov.
  - Problémy - výber vhodného úseku, vonkajšia fragmentácia, udržovanie informácie o voľných a obsadených úsekoch

Príklad (algoritmus plánovania FCFS, RR (q=1))



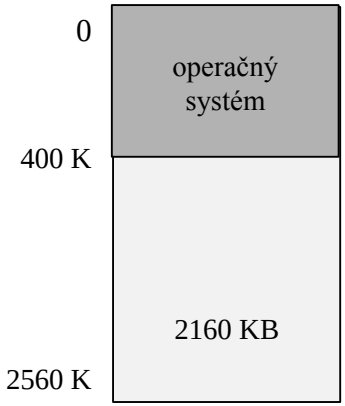
front pripravených procesov

proces	požadovaná pamäť	požadovaný čas
P_1	600 KB	10
P_2	1000 KB	5
P_3	300 KB	20
P_4	700 KB	8
P_5	500 KB	15



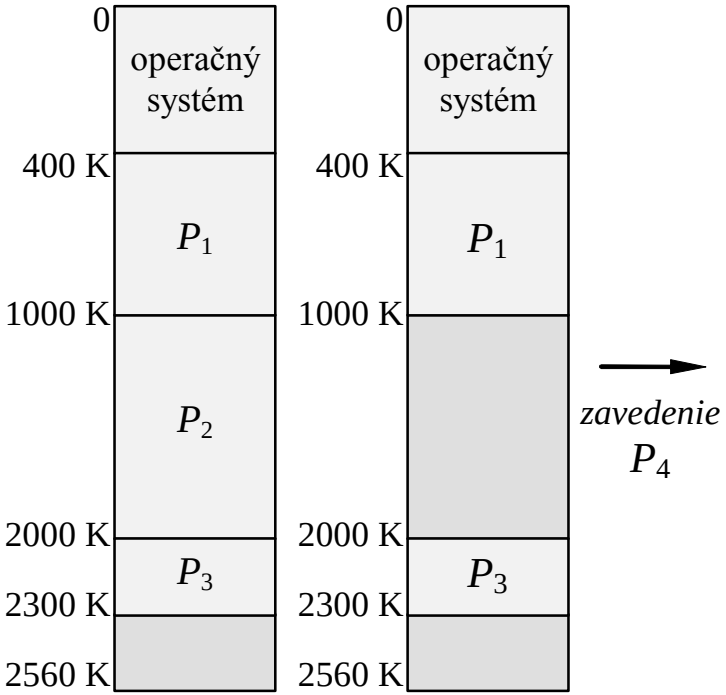
(a) FCFS

Príklad (algoritmus plánovania FCFS, RR (q=1))



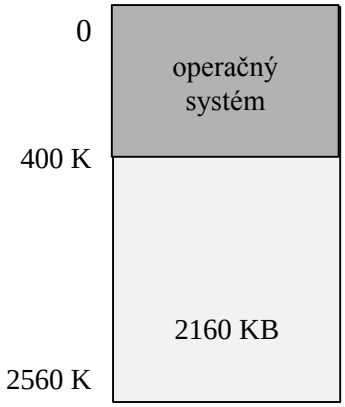
front pripravených procesov

proces	požadovaná pamäť	požadovaný čas
P_1	600 KB	10
P_2	1000 KB	5
P_3	300 KB	20
P_4	700 KB	8
P_5	500 KB	15



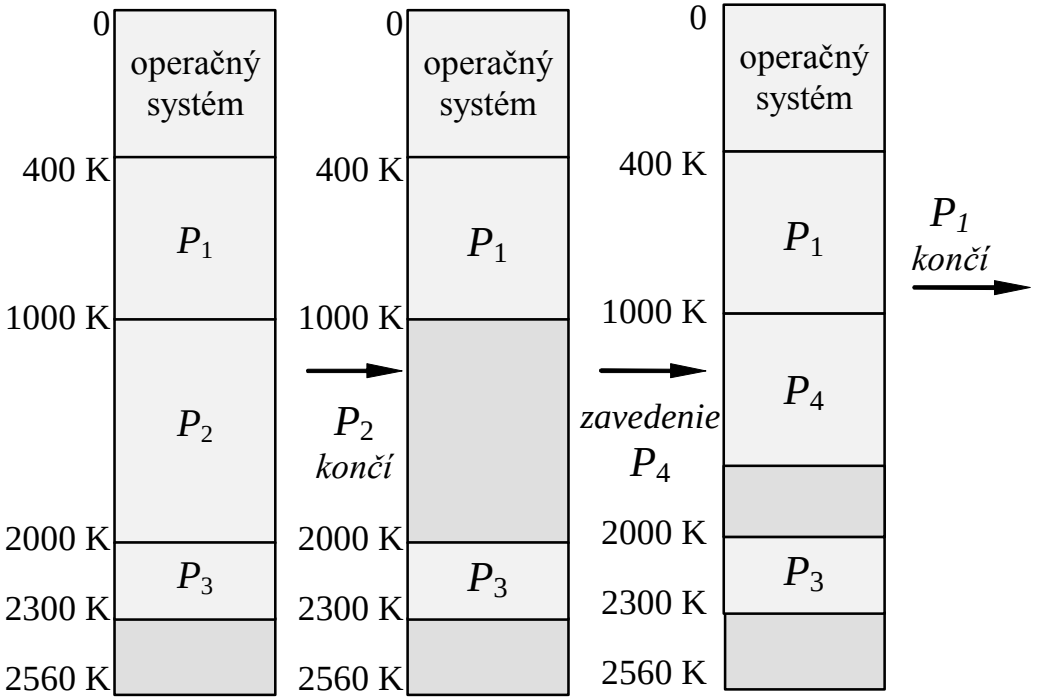
(a) FCFS (b) RR, q=1

Príklad (algoritmus plánovania FCFS, RR (q=1))



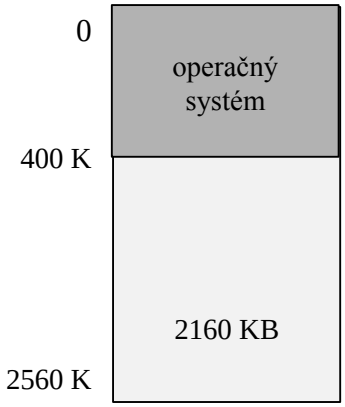
front pripravených procesov

proces	požadovaná pamäť	požadovaný čas
P_1	600 KB	10
P_2	1000 KB	5
P_3	300 KB	20
P_4	700 KB	8
P_5	500 KB	15



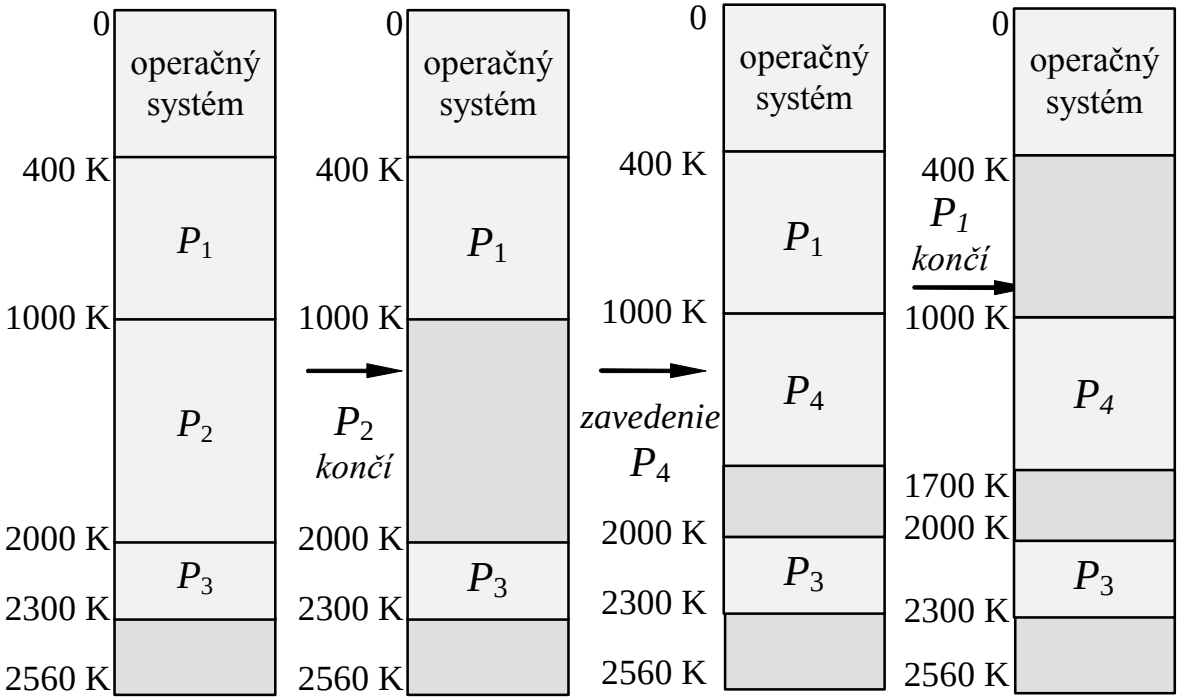
(a) FCFS (b) RR, q=1 (c)

Príklad (algoritmus plánovania FCFS, RR (q=1))



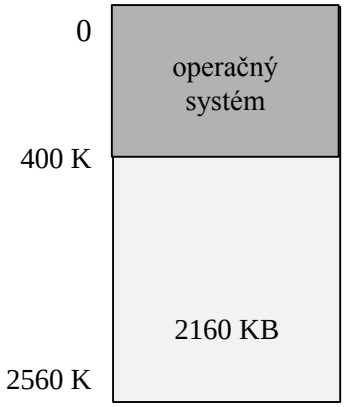
front pripravených procesov

proces	požadovaná pamäť	požadovaný čas
P_1	600 KB	10
P_2	1000 KB	5
P_3	300 KB	20
P_4	700 KB	8
P_5	500 KB	15



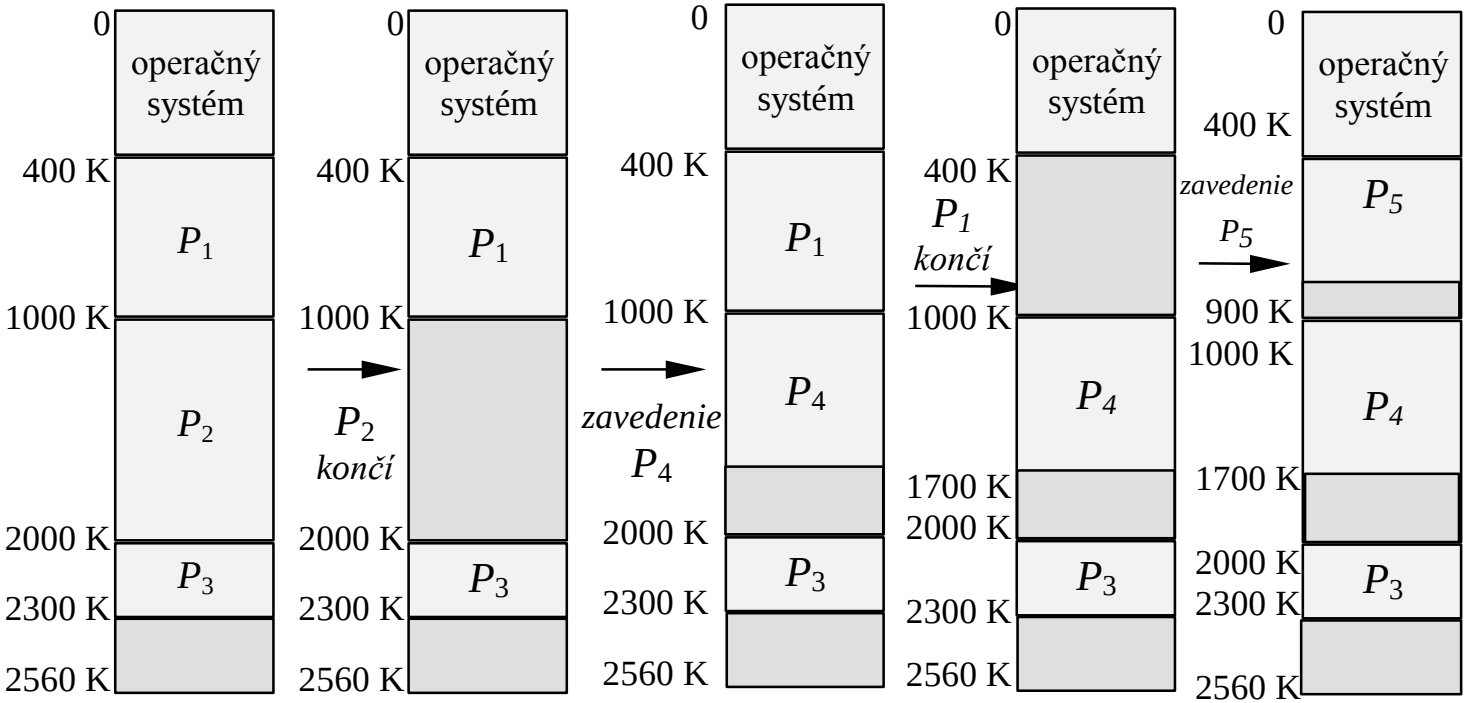
(a) FCFS (b) RR, q=1 (c) (d)

Príklad (algoritmus plánovania FCFS, RR (q=1))



front pripravených procesov

proces	požadovaná pamäť	požadovaný čas
P_1	600 KB	10
P_2	1000 KB	5
P_3	300 KB	20
P_4	700 KB	8
P_5	500 KB	15



Prideľovanie súvislých úsekov s premenlivou dĺžkou

pokračovanie

- Algoritmy výberu vhodného úseku pre umiestnenie procesu

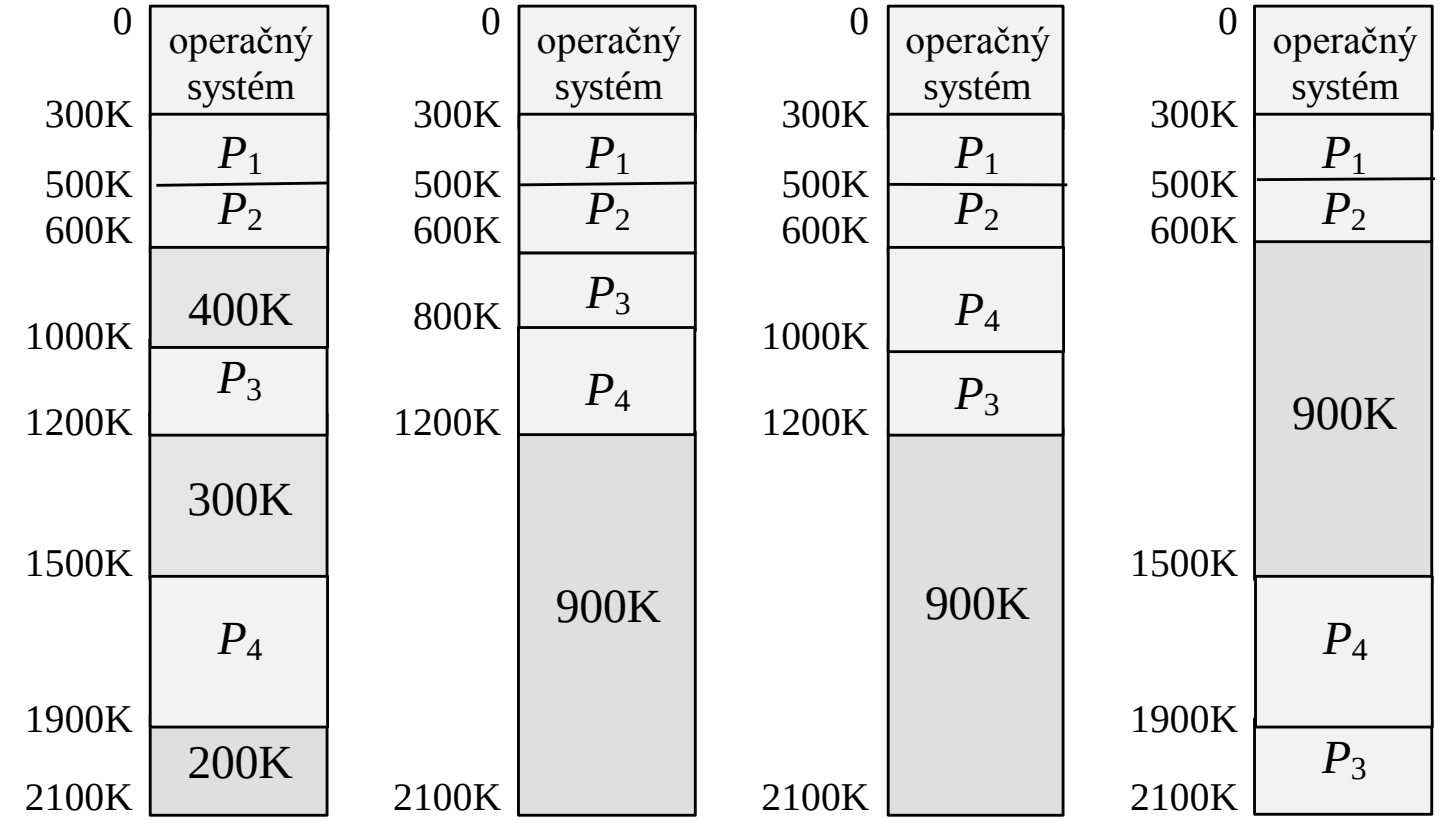
  Prvý vhodný (First-fit)

  Najlepšie vyhovujúci (Best-fit)

  Najhoršie vyhovujúci (Worst-fit)

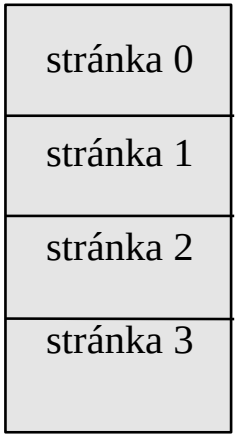
- Problém fragmentácie - vonkajšia
- Striasanie
  - spojiť dohromady fragmenty do jedného väčšieho bloku, príklad - riešenie situácie (e) (str.18) je na str. 20.
  - rôzne varianty striasania - na str.20

Porovnanie niekoľkých rôznych spôsobov kompresie pamäte

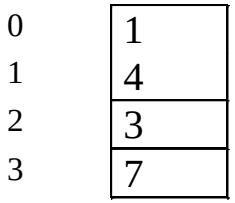


Stránkovanie

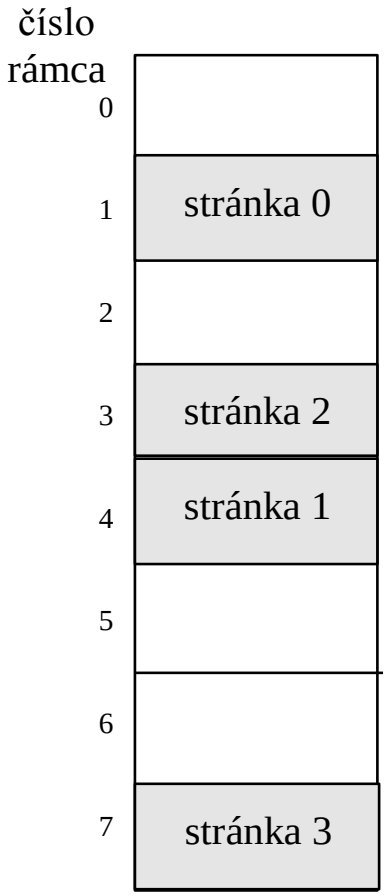
- Princíp
  - fyzická pamäť rozdelená na časti s pevnou veľkosťou - ***rámce***.
  - Logický adresný priestor procesu je rozdelený na rovnako veľké bloky, nazvané ***stránky***.



Logická pamäť



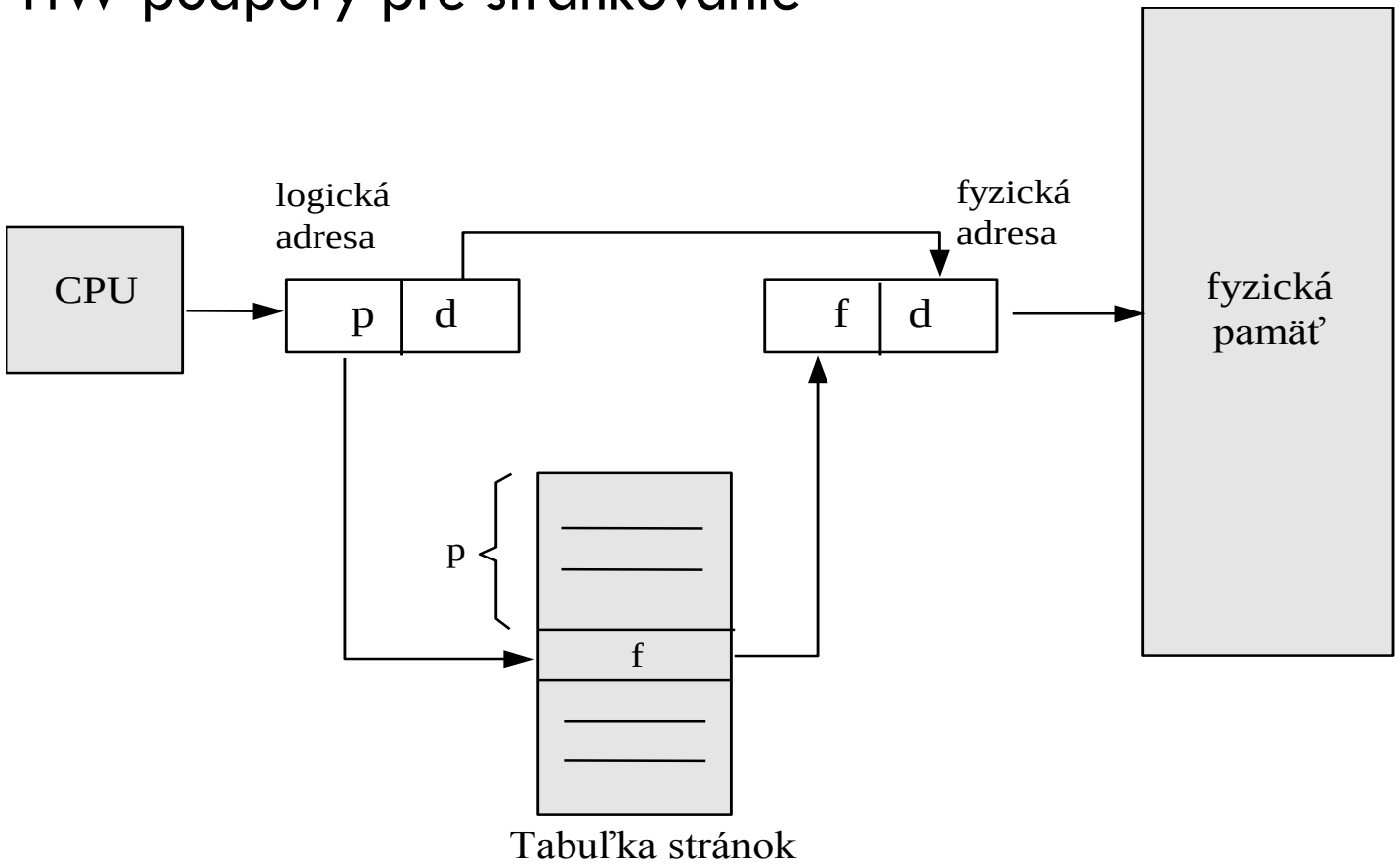
Tabuľka stránok



Fyzická pamäť

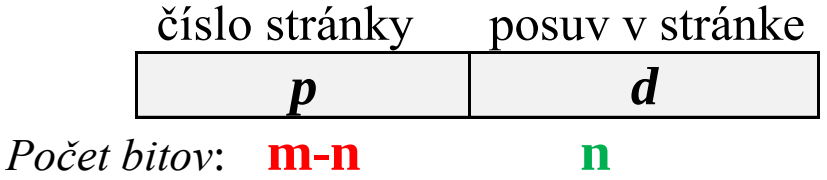
Stránkovanie pokračovanie

Potreba HW podpory pre stránkovanie



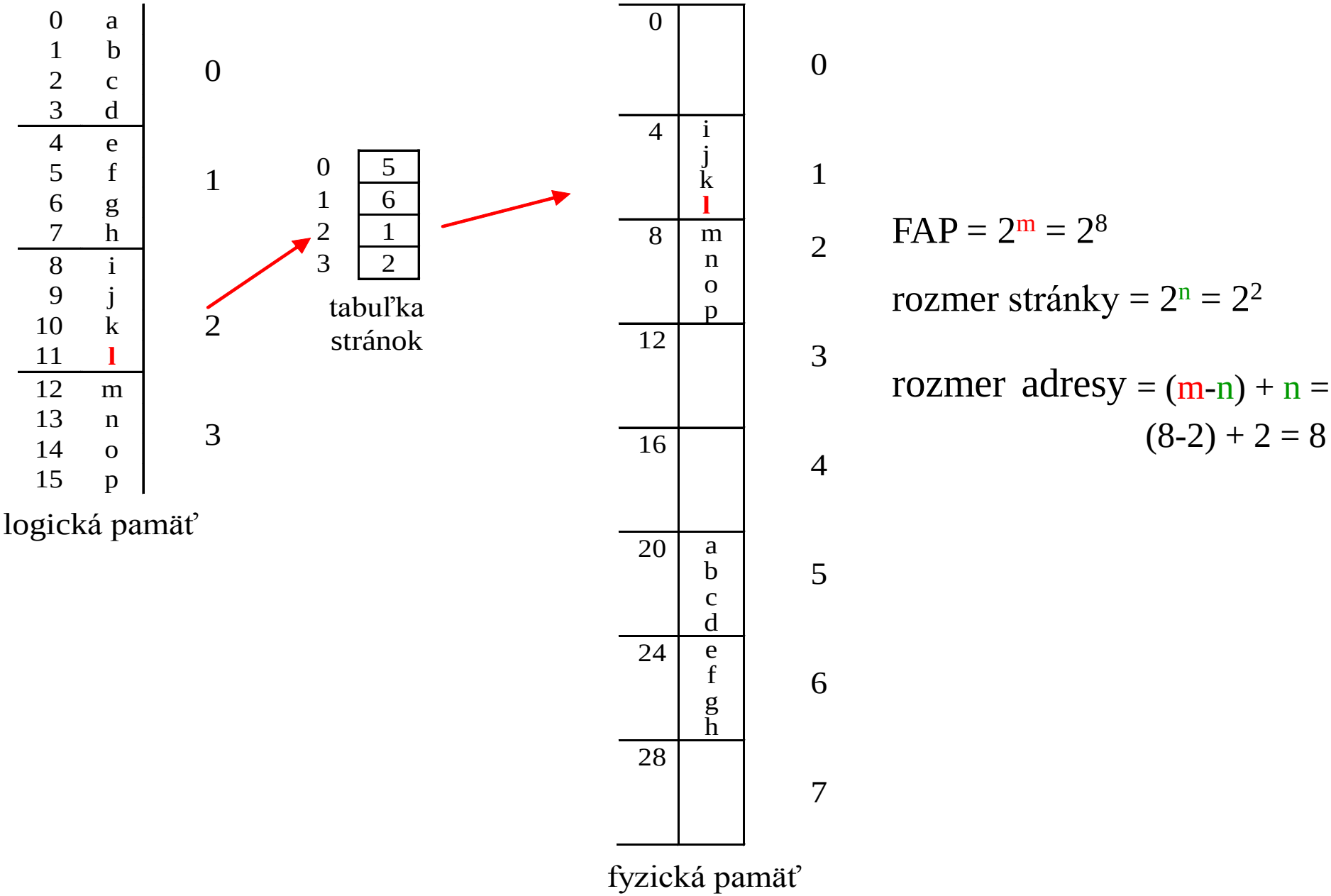
Strankovanie pokračovanie

– Rozmer stránky
  - obyčajne mocnina 2, od 512 bajtov do 8 KB
  - ak rozmer FAP= 2^{m} a veľkosť stránky = 2^{n} (bajtov alebo slov), potom
    – ***n*** nižších bitov - posuv v stránke.
    – vyššie ***m-n*** bity logickej adresy - číslo stránky



kde ***p*** - index do tabuľky stránok
d - posuv v stránke

Príklad stránkovania pre 32-bajtovú pamäť so 4-bajtovou stránkou



FAP = $2^m = 2^8$

rozmer stránky = $2^n = 2^2$

rozmer adresy = (m-n) + n =
(8-2) + 2 = 8

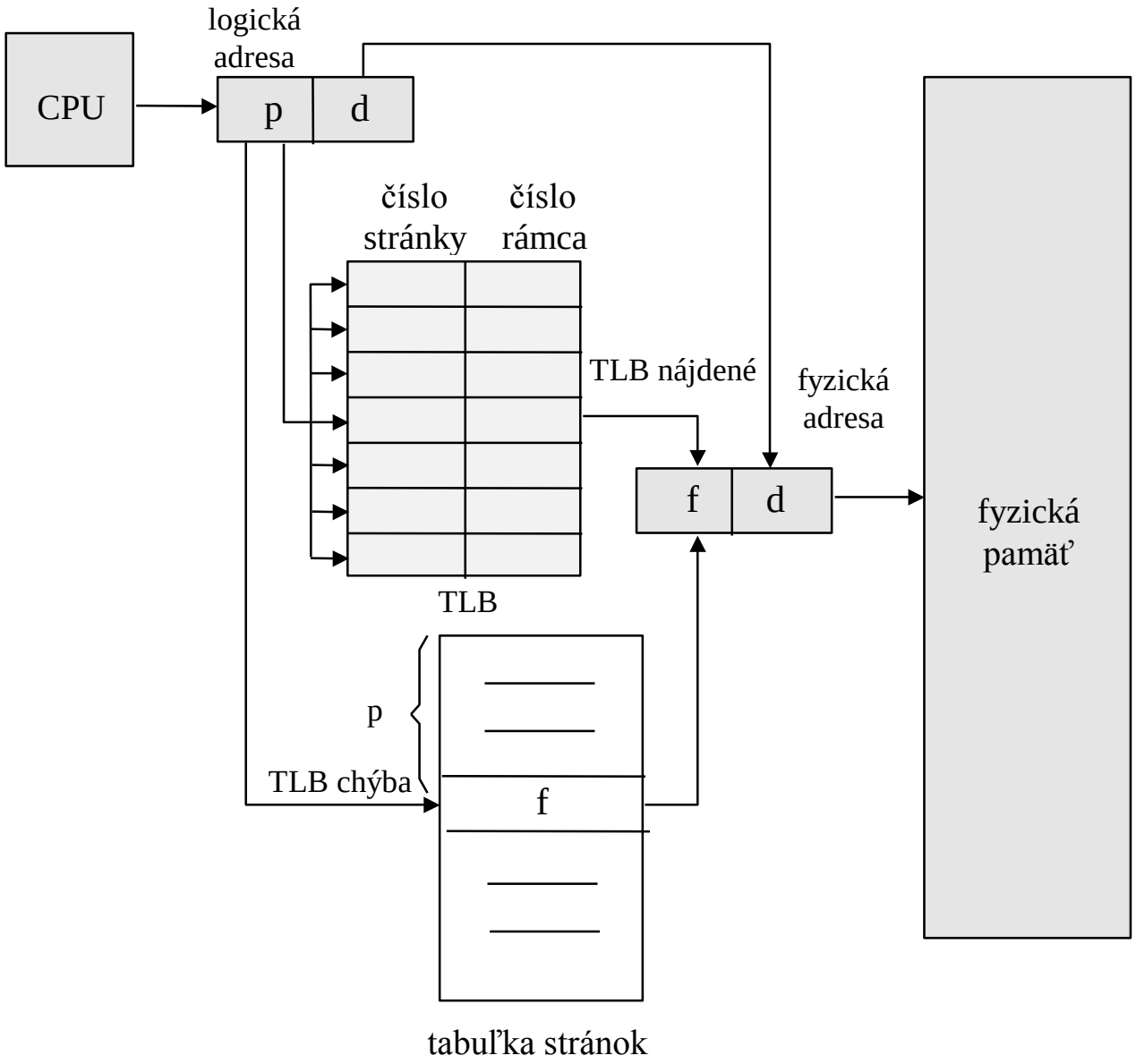
Stránkovanie pokračovanie

- Tabuľka rámcov
  - Informácie o voľných rámcov
- Tabuľka stránok
  - HW podpora
    - tabuľka stránok - v sade registrov
    - tabuľka stránok v pamäti - jej adresa sa nachádza v *registri tabuľky stránok* (Page-table base register, *PTBR*), dvojnásobný počet prístupov do pamäte

Stránkovanie pokračovanie

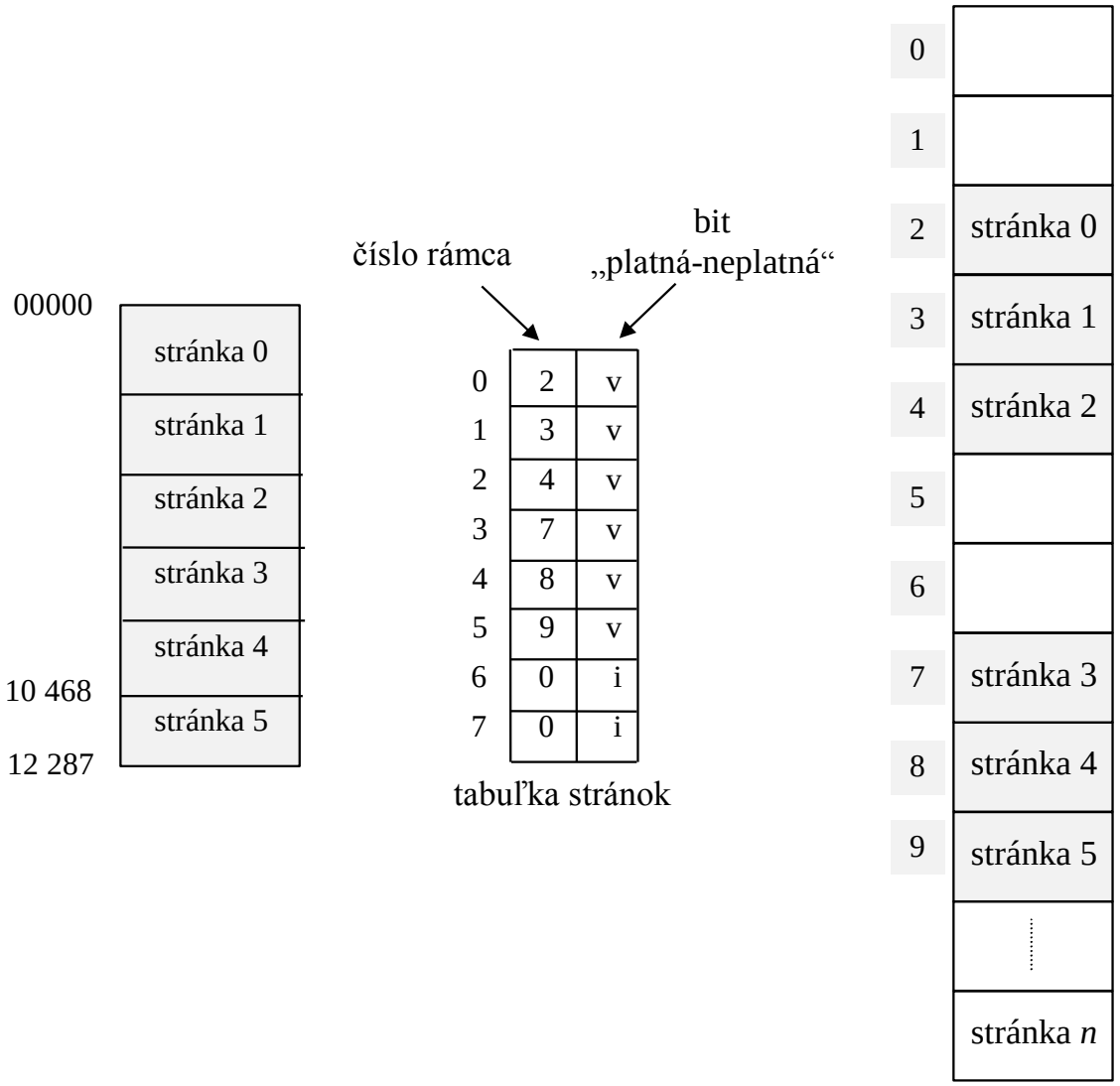
- Asociatívna cache pamäť (Translation Look-aside Buffers - TLB)
  - počet položiek od 8 do 2048
  - asociatívne prehľadávanie, hit rate (úspešnosť)
  - algoritmy nahradzovania
- Pri prepínaní kontextov procesov sa TLB musí vyčistiť
- Model stránkovania s TLB je na nasledujúcom obrázku

Stránkovací HW s TLB



Stránkovanie pokračovanie

- **Ochrana pamäte pri stránkovaní**
  - pomocou bitov, ktoré sú pripojené ku každému rámcu, **kontrolujú sa pri každom odkaze** (čítanie, zápis)
  - **bit *platná*/*neplatná*** (valid/invalid)



Viacúrovňové stránkovanie

- Viacúrovňové stránkovanie
  - súčasné systémy - veľmi veľký logický adresný priestor (od 2^{32} po 2^{64}) –> extrémne veľká tabuľka stránok.
  - Napr.

    pre systém s 32 adresou a stránkou 4 KB (2^{12} bajtov)

    tabuľka stránok - 1 000 000 položiek ($2^{32}/2^{12}$)

    → pre tabuľku stránok 4 MB pamäte

Riešenie – rozdeliť tabuľku stránok na stránky

Viacúrovňové stránkovanie pokračovanie

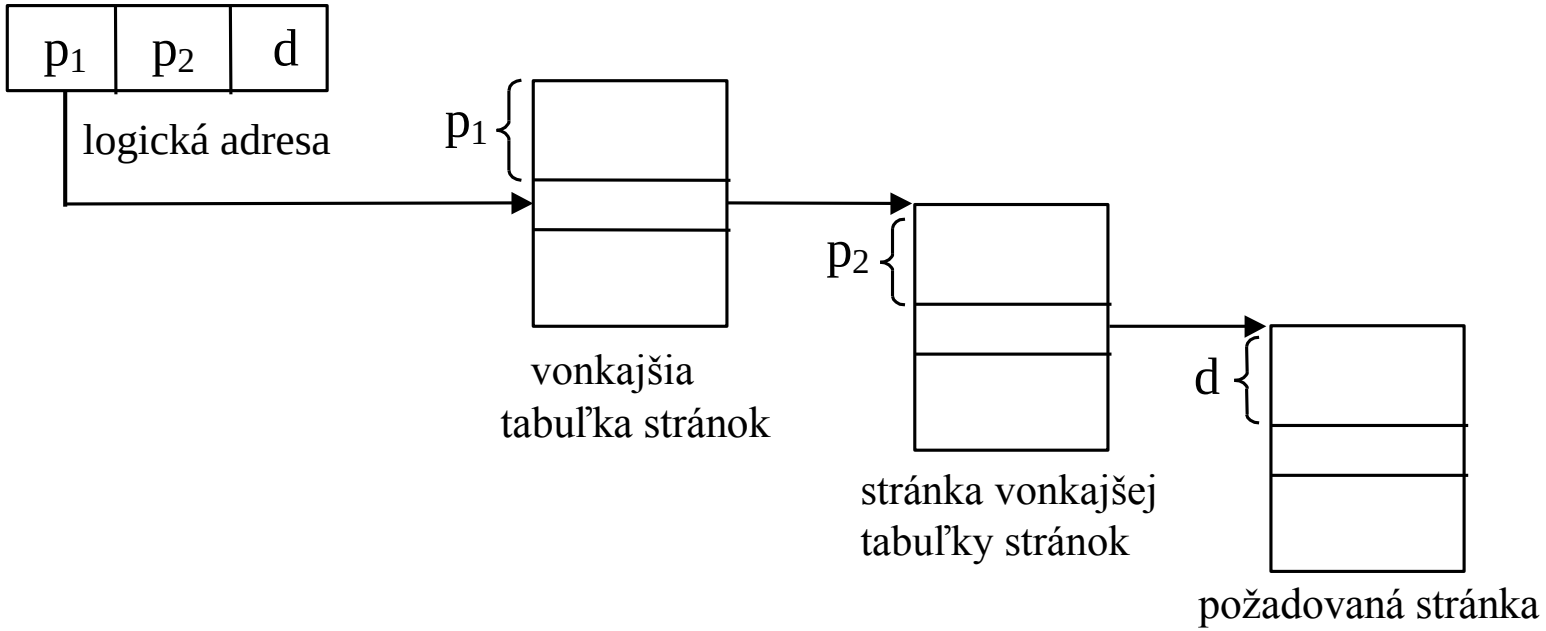
- Logická adresa :

číslo stránky		posuv v stránke
P_1	P_2	d
10	10	12

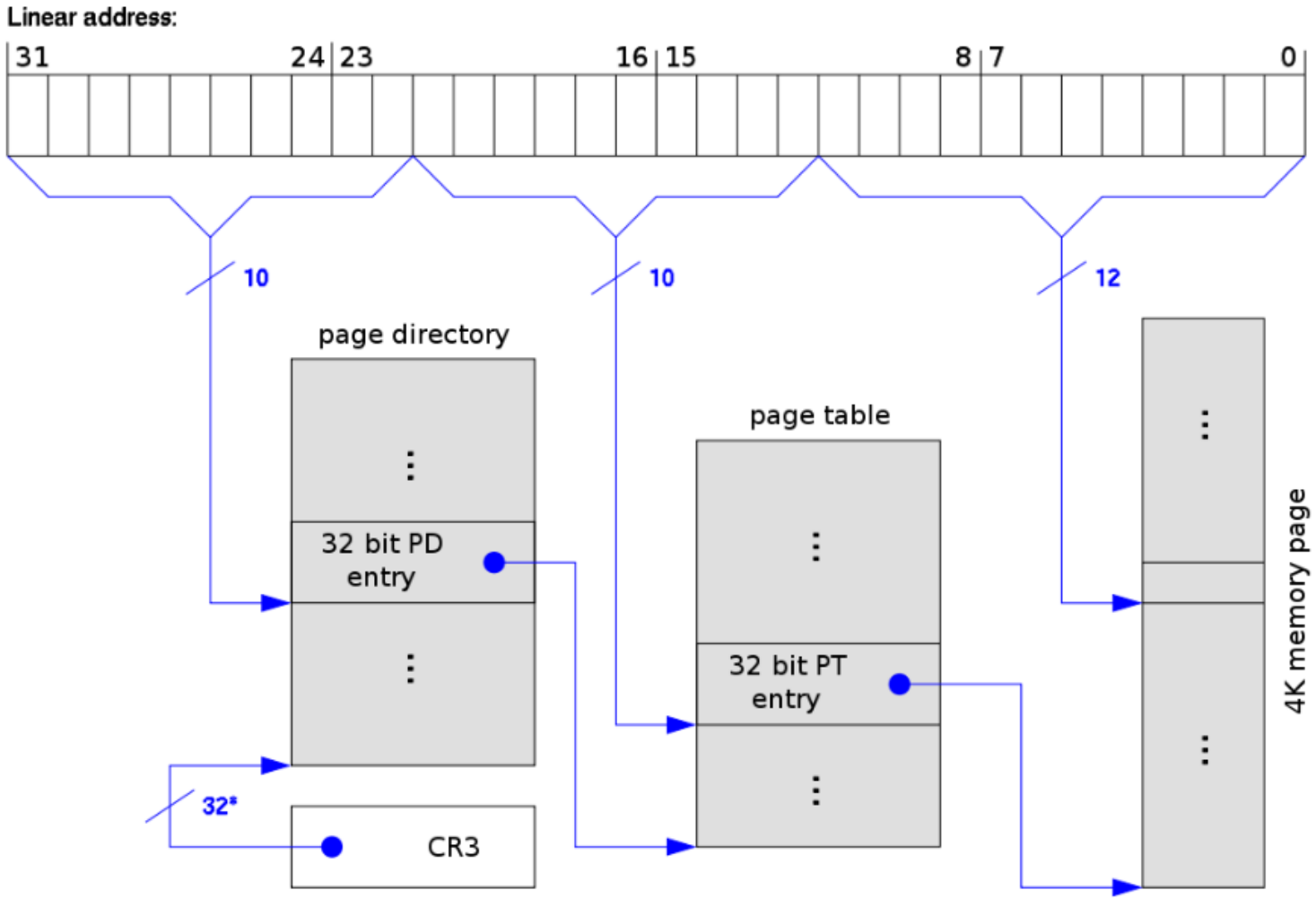
kde P_1 - index do vonkajšej tabuľky stránok
P_2 - posuv v stránke vonkajšej tabuľky

Príklad: nasledujúci obrázok

Prevod adries pre dvojúrovňovú 32 bitovú stránkovaciu architektúru

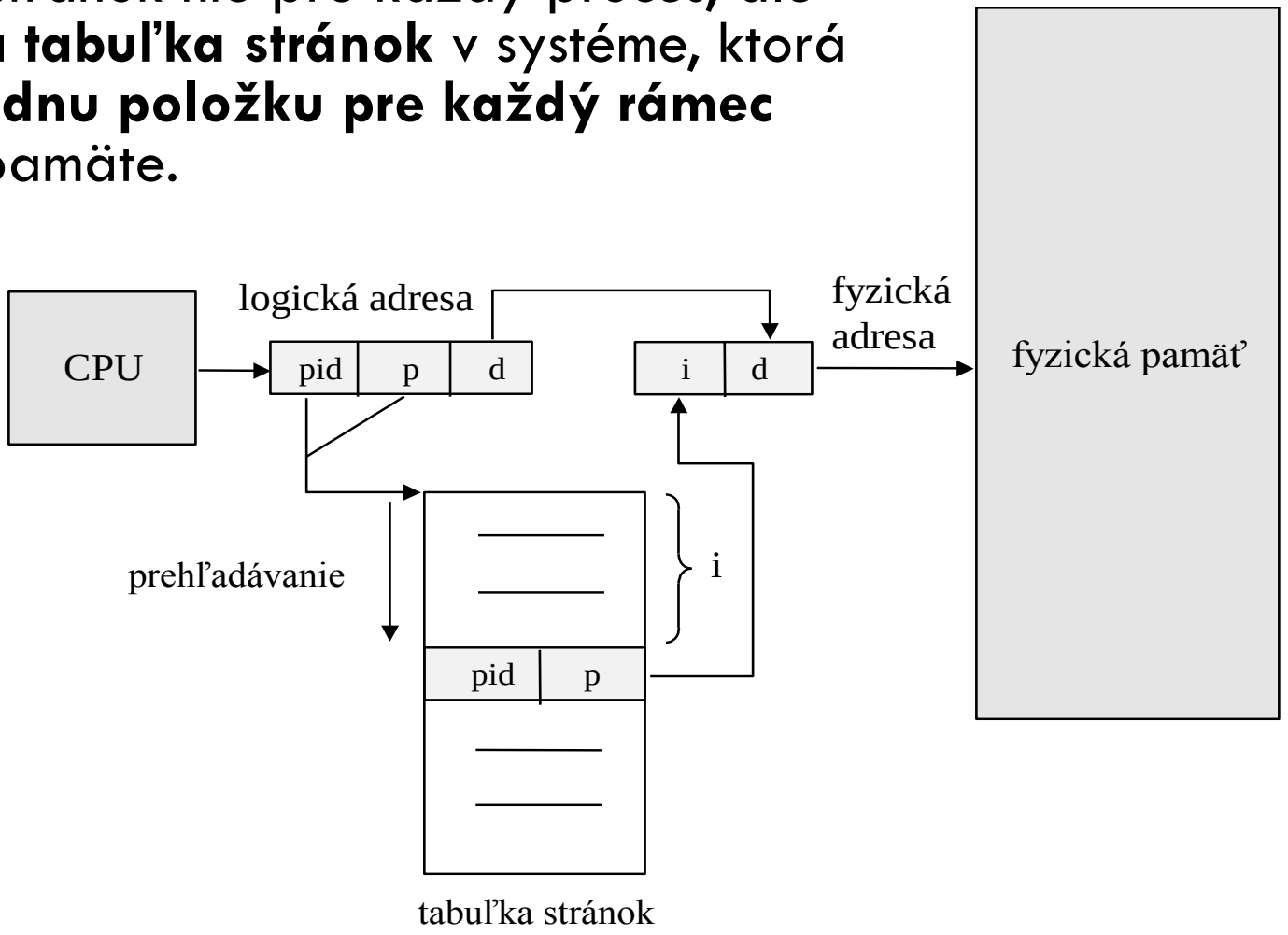


Prevod adries pre dvojúrovňovú 32 bitovú stránkovaciu architektúru – príklad



Invertovaná tabuľka stránok

- Tabuľka stránok nie pre každý proces, ale **len jedna tabuľka stránok** v systéme, ktorá má **len jednu položku pre každý rámec** fyzickej pamäte.



Invertovaná tabuľka stránok pokračovanie

- Hashovanie na základe PID procesu a čísla jeho rámca pre urýchlenie prehľadávania
- Indexy posledne použitých položiek môžu byť v TLB
- Invertovaná tabuľka stránok je využitá v systémoch
  - PowerPC,
  - UltraSPARC a
  - IA-64
- Problém
  - pri zdieľaní stránok – ako odvodiť rovnaký index v tabuľke z rôznych PID a čísel rámcov?

Zdieľanie stránok

- ***Reentrantný kód* - kód, ktorý môže byť bezpečne vykonávany paralelne**

Nereentrantný kód

```
int g_var = 1; // globálna premenná

int f() {
    g_var = g_var + 2;
    return g_var; }

int g() { return f() + 2; }
```

Reentrantný kód

```
int f(int i) { return i + 2; }

int g(int i) { return f(i) + 2; }
```

Zdieľanie stránok pokr.

- Aby bol kód funkcie *reentrantný*, musí:
  - Pracovať **len** s dátami, poskytnuté vo volaní alebo s kópiami globálnych dát
  - **Nesmie** uchovávať globálne dáta medzi jednotlivými volaniami
  - Nesmie vrátiť ukazovateľ na globálne dáta
  - **Nesmie volať** nereentrantnú funkciu.

Ochrana pamäte pri stránkovaní

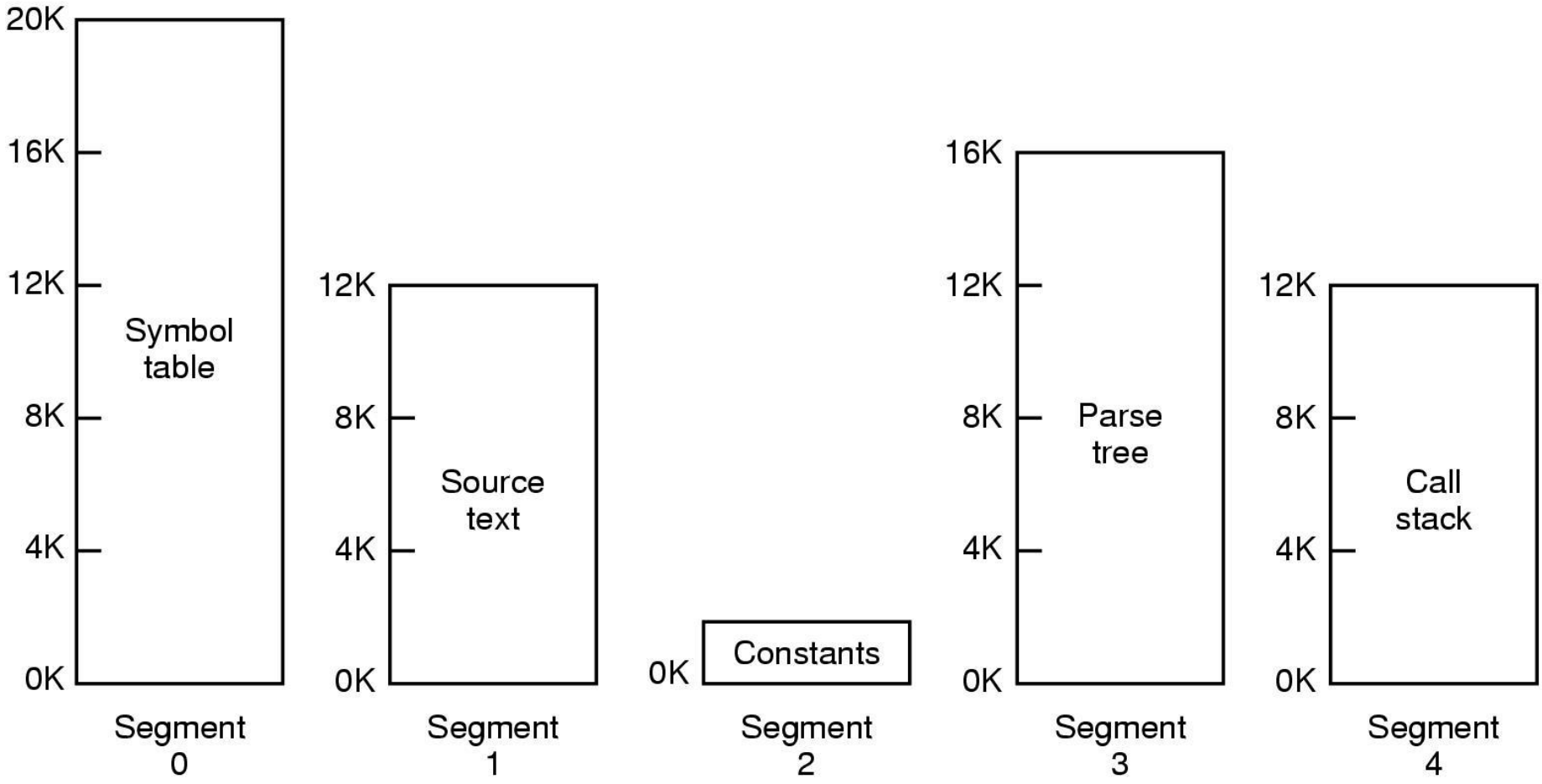
- Špeciálne bity (HW)
  - ***režim prístupu***
    - read, write, execute
  - ***valid/invalid*** – pre zistenie či stránka patrí do adresného priestoru procesu
    - môže byť poskytnutý ***page-table length register***, ktorý „odreže“ nepoužité stránky z tabuľky
    - užitočný, ak proces využíva veľmi malú časť z adresného priestoru.

Segmentácia

- Princíp
  - Pohľad používateľa na pamäť :

    segmenty s variabilnou dĺžkou
  - Logický adresný priestor je sada segmentov.
- Každý segment má
  - začiatok,
  - veľkosť.

Segmentácia - príklad



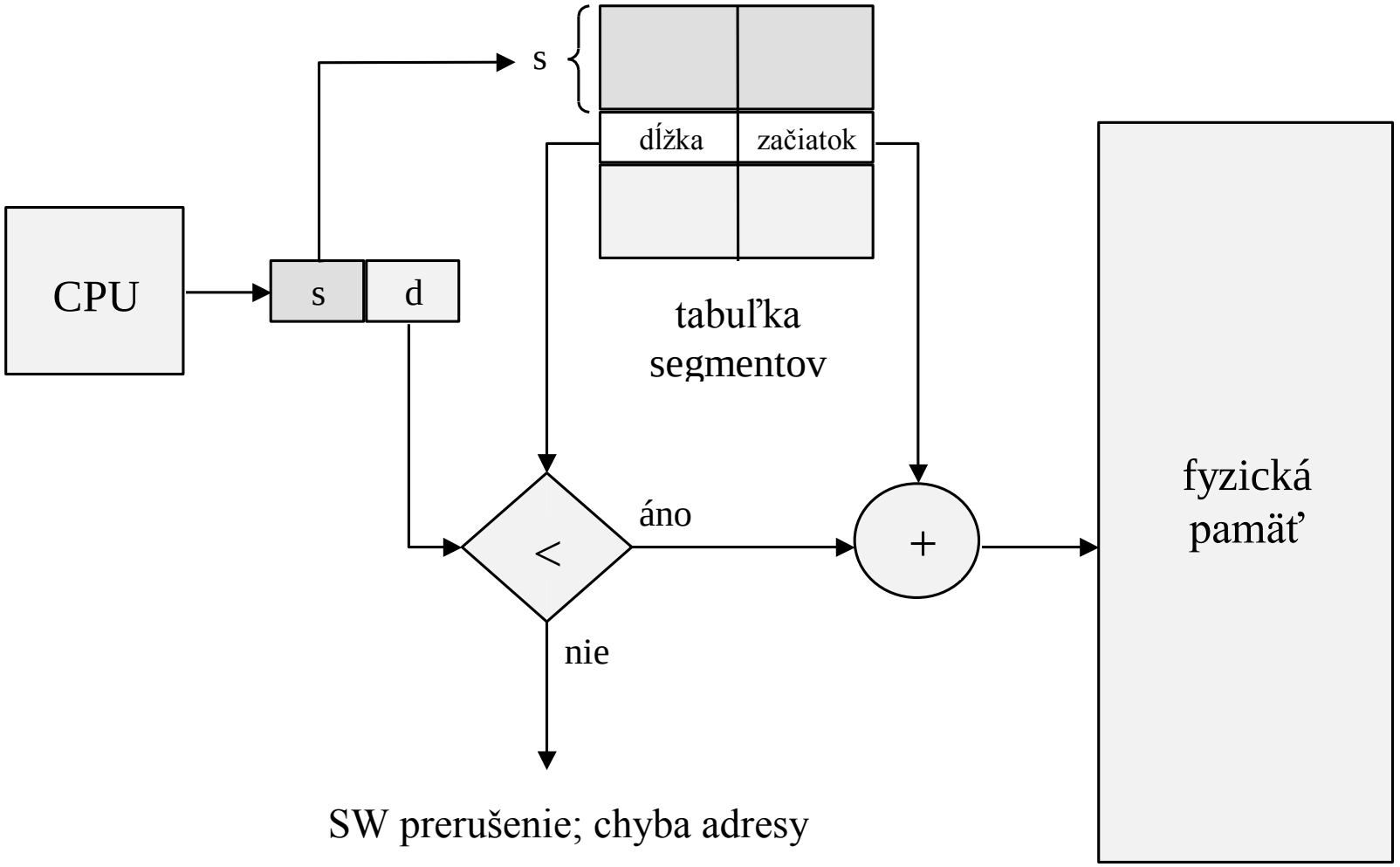
Segmentácia pokračovanie

- Logická adresa pozostáva z čísla segmentu a posuvu v segmente:

  <číslo segmentu, posuv>

- Keď sa program prekladá, prekladač automaticky vytvára segmenty:
  - globálne premenné, zásobník pre volania podprogramov a pre uloženie parametrov a návratových adries, kódy procedúr a funkcií a lokálne premenné každej procedúry a funkcie

HW podpora pri segmentácii



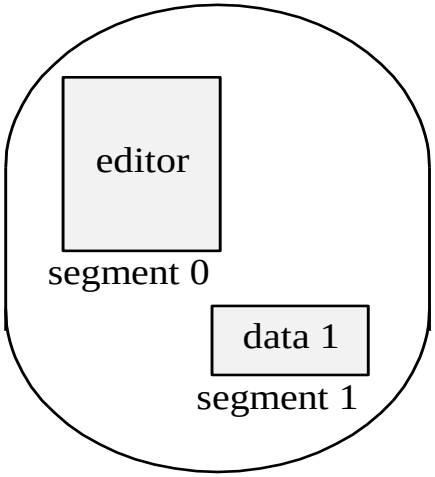
Implementácia tabuľky segmentov

- Mapovanie logických adries - cez tabuľku segmentov
- Tabuľka segmentov sa môže umiestniť
  - **do rýchlych registrov** - dá sa spracovať veľmi rýchlo
  - **do pamäte** – v prípade väčšieho počtu segmentov
    - HW podpora
      - ***Register začiatku tabuľky segmentov*** (STBR - Segment-Table Base Register)
      - ***Register dĺžky tabuľky segmentov*** (STLR - Segment-Table Length Register)
  - Pre urýchlenie prístupu do ST - sady asociatívnych registrov, ktoré uchovávajú posledne používané položky z tabuľky segmentov.
    - môže zredukovať čas pre prístup k pamäti tak, že ten **nebude o viac ako 10 až 15 % väčší** ako nemapovaný prístup.

Ochrana a zdieľanie

- Ochrana
  - ku každému segmentu – bity *read, write, execute*
- Zdieľanie nasledujúci obrázok
  - na úrovni segmentov
  - Problémy pri zdieľaní
    - kódové segmenty obsahujú odkazy sami na seba
    - riešenie - **register čísla segmentu**

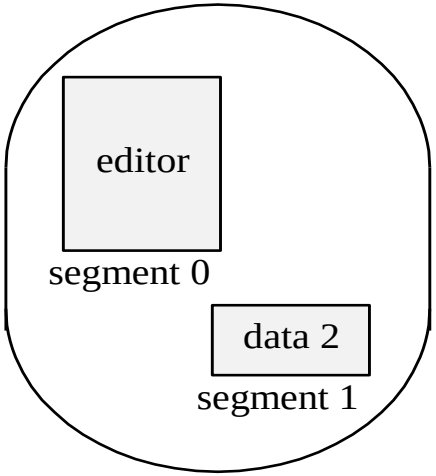
Zdieľanie segmentov



logická pamäť procesu P_1

	dĺžka	začiatok
0	25 286	43 062
1	4 425	68 348

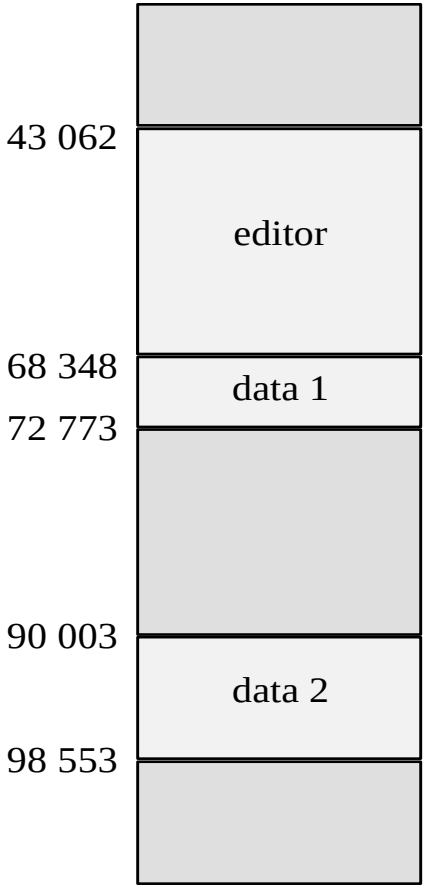
tabuľka segmentov procesu P_1



logická pamäť procesu P_2

	dĺžka	začiatok
0	25 286	43 062
1	8 850	90 003

tabuľka segmentov procesu P_2



fyzická pamäť

Fragmentácia

- Segmentácia spôsobuje **vonkajšiu** fragmentáciu
- Algoritmy pre prehľadávanie voľných úsekov
  - first-fit
  - best-fit.

Segmentácia so stránkovaním

- Najrozšírenejšia rady procesorov
  - Motorola 6800 - jednorozmerný adresný priestor t.j. stránkovanie
  - Intel 80X86 segmentácia
- **Segmentácia so stránkovaním**
  - navrhnuté pre GE 645 / Multics
  - Intel X86 používa segmentové registre na generovanie 32-bitovej logickej adresy, ktorá sa transformuje na fyzickú (prípadne aj viacúrovňové stránkovanie)

Stránková segmentácia na GE 645 (MULTICS)