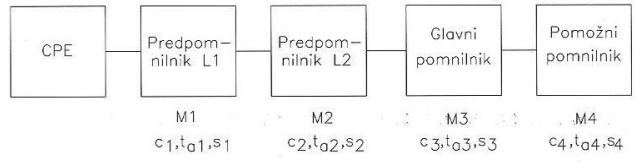
# 11
# NAVIDEZNI POMNILNIK

## Navidezni pomnilnik

- Razlog za pojav NP je želja po povečanju GP
- Navidezni pomnilnik (virtual memory) je prostor v pomožnem pomnilniku (magnetni disk), ki je za uporabnika videti kot GP
  - to zahteva dodatno logiko v CPE
    - običajen dostop do pomožnega pomnilnika poteka sicer preko V/I ukazov (tj. bistveno drugače od dostopa do GP)
    - mehanizem naredi te razlike nevidne
- Vse skupaj je del pomnilniške hierarhije

# Pomnilniška hierarhija

- PH je zaporedje pomnilnikov ($M_1$, $M_2$, ..., $M_n$), pri katerem vsak pomnilnik komunicira samo s svojima sosedoma




- Vsebina $M_{i-1}$ je podmnožica vsebine $M_i$
- Celotna informacija je le na $M_n$
- CPE vidi PH kot en sam pomnilnik
  - naslov, ki ga da CPE, je naslov besede na najvišjem nivoju ($M_n$)
  - vendar hitrejši



- PH funkcionira le zaradi lokalnosti pomnilniških dostopov
- PH prinese tudi probleme: čas dostopa spremenljiv in težko napovedljiv (običajno le statistično)

cena / bit: $c_i > c_{i+1}$

dostopni čas: $t_{ai} > t_{a,i+1}$

velikost: $s_i < s_{i+1}$

N dostopov:
- v $N_1$ primerih je inf. v $M_1$,
- v $N_2$ primerih je inf. v $M_2$ in ne v $M_1$,
- v $N_3$ primerih je inf. v $M_3$ in ne v $M_1$ ali $M_2$,
- itd.

$$N = N_1 + N_2 + \ldots N_n$$

Globalna verjetnost zadetka

$H_1 = N_1/N$
$H_2 = (N_1 + N_2)/N$
...
$H_n = (N_1 + N_2 + \ldots + N_n)/N = 1$

Ekskluzivna verjetnost zadetka

$h_i = N_i/N$

to ni lokalna verj. zadetka (le-ta je $N_i/(N - N_1 - N_2 - \ldots - N_{i-1})$ )

$h_1 = H_1$
$h_i = H_i - H_{i-1}$, i=2, 3, ..., n

torej:

$h_2 = H_2 - H_1$,
$h_3 = H_3 - H_2$, ...



Cena bita

$$c = (c_1s_1 + c_2s_2 + \ldots + c_ns_n) / (s_1 + s_2 + \ldots + s_n)$$

Če želimo, da bo c blizu $c_n$, mora biti $s_n$ veliko večji od vseh ostalih skupaj.

Povprečen čas dostopa, kot ga vidi CPE:
Če informacije ni na nivojih pod i, se mora prenesti z nivoja i na nižje nivoje.
Čas dostopa do nivoja i je

$$T_i = \sum_{k=1}^{i} t_{ak} = t_{a1} + \cdots + t_{ai}$$

Verjetnost za to je $h_i$.

Povpr. čas dostopa n-nivojske PH:

$$\begin{aligned} t_a = \textstyle\sum_{i=1}^{n} h_i T_i &= h_1T_1 + h_2T_2 + \cdots + h_nT_n \\ &= h_1t_{a1} + h_2(t_{a1} + t_{a2}) + \cdots + h_n(t_{a1} + \cdots + t_{an}) \\ &= t_{a1} \cdot 1 + t_{a2} \cdot (1 - H_1) + \cdots + t_{an} \cdot (1 - H_{n-1}) \\ &= t_{a1} + \textstyle\sum_{i=2}^{n} (1 - H_{i-1}) \cdot t_{ai} \end{aligned}$$

Če imamo samo 2 nivoja:

$$t_a = t_{a1} + (1-H_1)*t_{a2}$$

Bolj pravilno je

$$t_a = t_{a1} + (1-H_1)*t_B \qquad (t_B \text{ je čas prenosa bloka})$$

Lahko sta $M_1$ PP in $M_2$ GP, lahko pa sta $M_1$ GP in $M_2$ HD.
Zgrešitvena kazen pri PP je 10-100, pri HD pa 10-100 milijonov!

Pri NP se zato uporabljajo drugačne rešitve kot pri PP:

- dovolj veliki bloki, da ni preveč prenosov
- čista asoc. preslikava (da je čimmanj zgrešitev)
- zamenjava blokov se opravlja programsko (ne strojno)
  - Večja počasnost prog. rešitve je zanemarljiva v primerjavi s časom dostopa do bloka na disku.
  - Poleg tega je možno uporabiti bolj zahtevne algoritme za izbiro bloka, ki naj se zamenja
    - že z majhnim zmanjšanjem zgr. kazni se počasnost prog. rešitve več kot odtehta

NP uporabljajo le pisanje nazaj (pisanje skozi ni uporabno)



Zaradi zgr. je povp. hitrost dostopa manjša, kot če NP ne bi imeli.

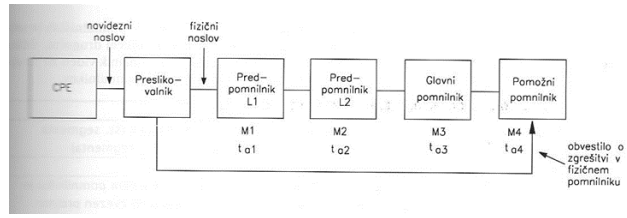
- Npr., da dovolimo 10% povečanje časa dostopa. Kolikšna sme biti največja verjetnost zgr. 1-H?
  - $t_{a1}$ = 40ns (DRAM), $t_B$ = 15ms (mag. disk)

$$t_a = t_{a1} + (1-H)*t_{a2} = 1{,}1*t_{a1}$$
$$(1-H)*t_{a2} = 0{,}1*t_{a1}$$
$$(1-H) = 0{,}1*t_{a1}/t_{a2} = 4\text{ns}/15\text{ms}$$
$$= 0{,}27*10^{-6} = 2{,}7*10^{-5}\,\%$$

Verj. zgr. je lahko največ $2{,}7*10^{-5}$ %, torej zelo majhna!

# Preslikovanje navideznih naslovov

- Naslovi, ki jih daje CPE, se pri vsaki PH vedno nanašajo na najvišji nivo.
- Pri rač. z NP se imenujejo **navidezni naslovi**
- Pri vsakem pomnilniškem dostopu je **navidezni naslov** potrebno preslikati v **fizični naslov** (v GP). Le-ta gre v PP.






Preslikovanje:

$A_f = f(A_n)$

$A_f$ ... fizični naslov

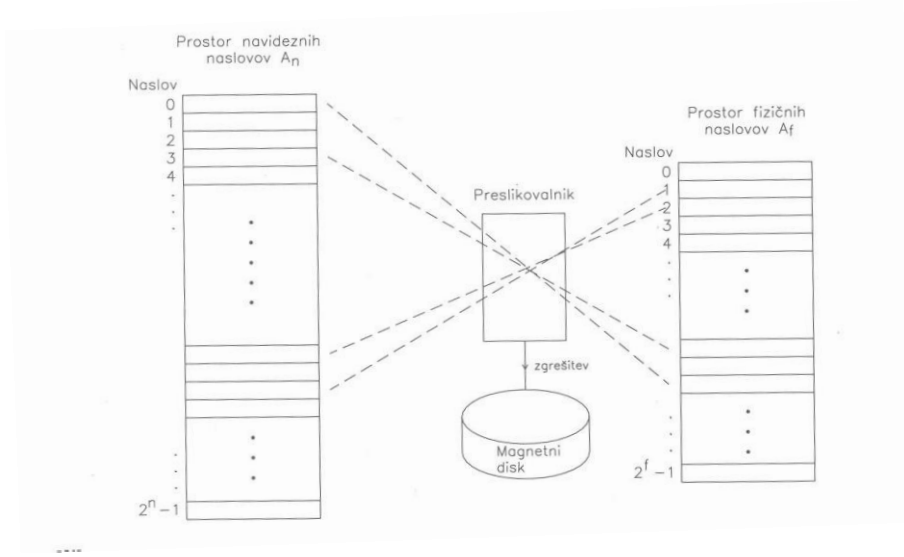
$A_n$ ... navidezni naslov

f ... preslikovalna funkcija

- f se sproti prilagaja, da je verj. zadetka čimvečja

Če je naslov $A_n$ prisoten v GP, ga preslikovalnik preslika v $A_f$

- če ni, preslikovalnik ugotovi zgrešitev in blok se prenese z diska v GP, nato se $A_n$ preslika v $A_f$ in CPE izvede dostop






- Preslikovanje izvaja **enota za upravljanje s pomnilnikom** (MMU, memory management unit)

- Pri NP je potrebna samo logika za preslikovanje, ne pa tudi za zamenjavo (kot pri PP)
  - zamenjevanje blokov je realizirano programsko
    - prekinitev, PSP prenese blok z diska

# Vrste navideznih pomnilnikov

2 vrsti:

- NP z bloki fiksne velikosti (**strani**)
  - Strani: 4, 8, 16KB (podobne so blokom v PP)
    - naslov je enodimenzionalen
    - zamenjava bloka preprosta (ker so vsi bloki enako veliki)
    - velikost strani lahko tudi prilagodimo lastnostim diska
- NP z bloki spremenljive velikosti (**segmenti**)
  - Segmenti so lahko različnih velikosti.



# Ostranjevanje

paging

- Pomožni pom. je razdeljen na bloke enake velikosti (**strani**, pages). Vse strani skupaj sestavljajo NP.
- GP je razdeljen na enako velike **okvire strani** (page frames).
- Vsako stran NP lahko prenesemo v poljuben okvir

Naj bo

- NP velikosti $2^n$ besed (npr. 8-bitnih)
- stran velikosti $2^p$ besed
- GP velikosti $2^f$ besed

število strani v NP = $2^n / 2^p = 2^{n-p}$

število okvirov v GP = $2^f / 2^p = 2^{f-p}$

| Navidezni naslov | Stran |
|---|---|
| 0<br>...<br>$2^p$-1 | 0 |
| $2^p$<br>...<br>$2*2^p$-1 | 1 |
| $2*2^p$<br>...<br>$3*2^p$-1 | 2 |
| .<br>.<br>. | |
| $(2^{n-p}-2)*2^p$<br>...<br>$(2^{n-p}-1)*2^p$ -1 | $2^{n-p}$-2 |
| $(2^{n-p}-1)*2^p$<br>...<br>$2^{n-p}*2^p$ -1 = $2^n$ - 1 | $2^{n-p}$-1 |

| Fizični naslov | Okvir strani |
|---|---|
| 0<br>...<br>$2^p$-1 | 0 |
| $2^p$<br>...<br>$2*2^p$-1 | 1 |
| | .<br>.<br>. |
| $(2^{f-p}-1)*2^p$<br>...<br>$2^{f-p}*2^p$ - 1 = $2^f$ - 1 | $2^{f-p}$-1 |



Preslikovalno funkcijo definira **tabela strani** (**page table**).

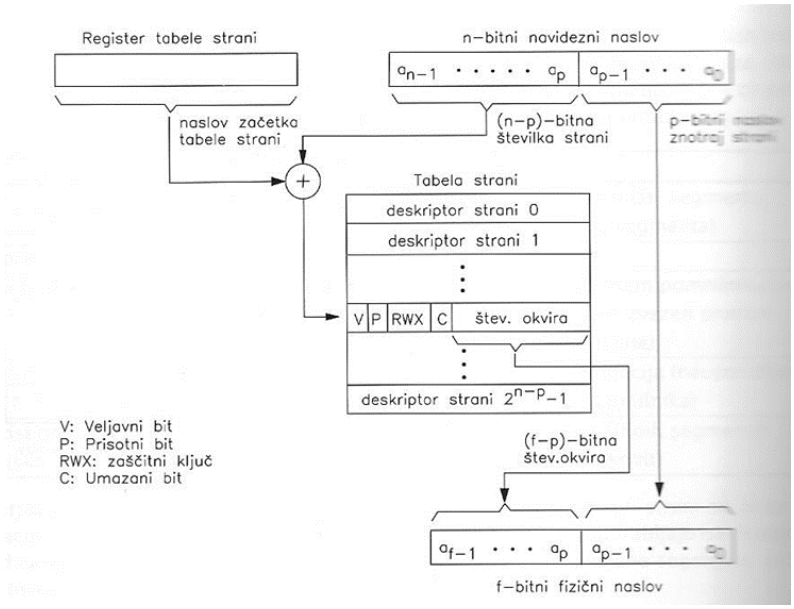
- Za vsako stran NP ima tabela eno polje (**opisnik strani**, **page descriptor**).

**Notranja fragmentacija**: vsak program zaseda določeno št. strani, v povp. pa ½ strani ostane neizkoriščena

Navidezni naslov ima 2 dela:

- zgornjih n-p bitov je **številka strani**
- spodnjih p bitov je **naslov besede znotraj strani** (ta po preslikavi ostane enak)

Številka strani se preko tabele strani preslika v številko okvira strani.






Opisnik strani ima 5 parametrov:

- **Veljavni bit V** (valid)
  - na začetku je 0
- **Prisotni bit P** (present) pove, ali je stran v GP
  - P=1: 'zadetek' (take strani se imenujejo **aktivne strani**)
    - FN pove, kje v GPju je stran
  - P=0: 'zgrešitev' (**napaka strani** (**page fault**))
    - FN ni veljaven
    - stran je treba prenesti z diska v GP
      - to naredi PSP, ko se sproži past
- **Umazani bit C** (change)
  - Ko se stran prenese v GP, se C postavi na 0
  - Če pride do pisanja na vsaj en naslov na tej strani, gre C na 1
  - Ko se stran zamenja z neko drugo, je treba stran prenesti na disk samo, če je C=1 (sicer je itak enaka kot na disku)

- **Zaščitni ključ RWX** pove vrsto dostopa do strani
  - read, write, execute
  - Namenjeno zaščiti pomnilnika
  - Vsak program ima svojo tabelo strani
  - Spreminjanje vsebine tabel strani je dovoljeno samo v privilegiranem načinu delovanja OS (le v takem načinu je dovoljeno pisati v del pom., kjer je tab. strani)
  - Kadar iz zašč. ključa sledi, da dostop ni dovoljen, se sproži past zaščite
    - ta ima višjo prioriteto kot napaka strani (bit P se ignorira)
- **Številka okvira FN** (Frame number) podaja številko okvira, v katerem je stran
  - naslov okvira je FN $\times$ $2^p$ (če je seveda P=1, sicer stran ni v GPju)
  - k FN $\times$ $2^p$ se prišteje naslov znotraj strani in dobimo fizični naslov besede, do katere se dostopa
  - tudi naslov okvira ima spodnjih p bitov enakih 0 (tako kot naslov strani v NP)



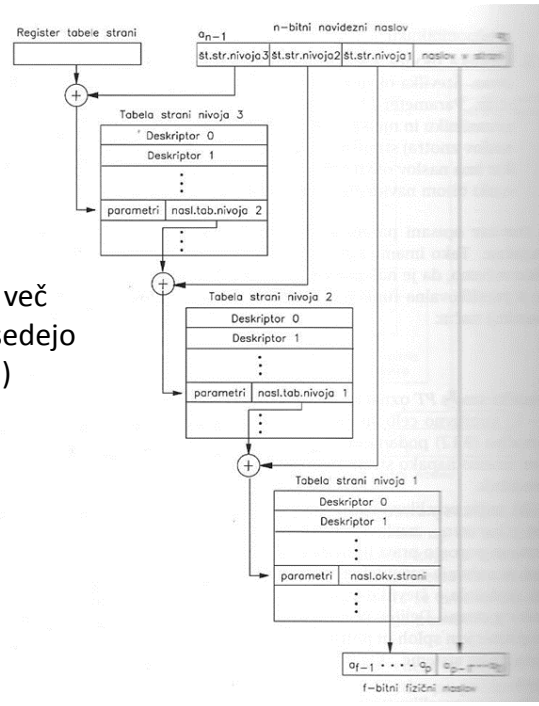
Preslikovalna funkcija *f*:

$A_f = FN * 2^p + A_n(p-1 : 0)$ PT ... tabela strani

$FN = PT[A_n(n-1 : p)]$ FN ... številka okvira iz PT

- Preslikovalna funkcija velja pri pogoju, da je v opisniku strani bit P enak 1. Sicer imamo napako strani in stran se najprej prenese v GP, šele nato se izvede preslikava.
- To je **linearno preslikovanje**.

Danes so PT v več nivojih (da zasedejo manj prostora)




21

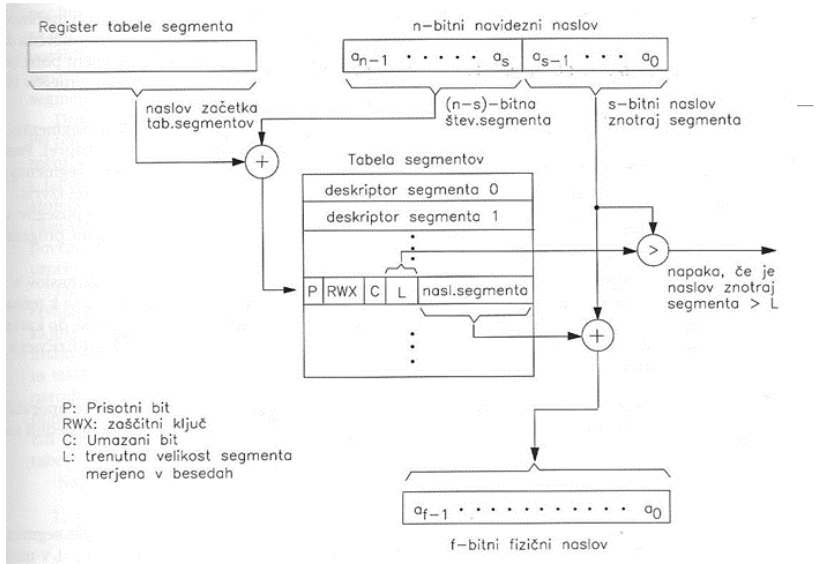
- **Premetavanje** (thrashing): Kadar je napak strani veliko, se lahko večji del časa prenašajo strani med GP in diskom, uporabniški programi pa čakajo.
- Ostranjevanje je enostavno, ne omogoča pa uporabe boljših načinov za zaščito pomnilnika, pa tudi več programov si ne more deliti istega fizičnega prostora (pri segmentaciji si lahko).

# Segmentacija

- Segmenti so različnih dolžin
- Vsebina segmenta ima za program svoj pomen
- Število in velikost segmentov se lahko med izvajanjem programa spreminjata
- Segment se lahko prenese na poljuben naslov v GP

- Zgornjih *n-s* bitov navideznega naslova določa **številko segmenta**, spodnjih *s* bitov pa **naslov besede v segmentu** (odmik, segment offset).
- V **tabeli segmentov** ima vsak segment svoj **opisnik segmenta** (ki ga naslovimo s številko segmenta + začetni naslov tabele), ki vsebuje tudi naslov segmenta.
- Fizični naslov dobimo tako, da seštejemo naslov segmenta in naslov besede v segmentu.





- Hiba je v tem, da je odmik treba prištevati – zato za prevajalnike oz. programerje ni neviden.

**Segmentacija z ostranjevanjem**

- Segmenti so razdeljeni na strani. Potrebna je tabela segmentov in tabele strani (po ena za vsak segment).
- Namen je izkoriščanje prednosti segmentacije, vendar brez problemov, ki se pojavijo pri čisti segmentaciji.



# Problemi pri realizaciji navideznega pomnilnika

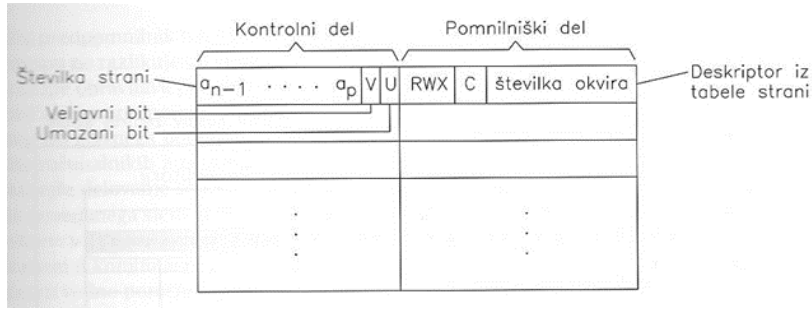
Problemov je več, najpomembnejši pa je, kako pospešiti preslikovanje.

Pri vsakem pom. dostopu sta potrebna 2 dostopa (pri več nivojih še več):

- Dostop do tabele strani v GP
- Dostop do fizičnega naslova

Računalnik bi postal prepočasen.

**Preslikovalni predpomnilnik** (translation cache) ali **TLB** (translation lookaside buffer) vsebuje podmnožico opisnikov, ki so bili nedavno uporabljeni.


Kontrolni del
Pomnilniški del
Številka strani
$a_{n-1}$ · · · · $a_p$
V
U
RWX
C
številka okvira
Deskriptor iz tabele strani
Veljavni bit
Umazani bit