Navidezno naslavljanje

⇓

Enota za upravljanje s pomnilnikom (MMU)

Želimo si:

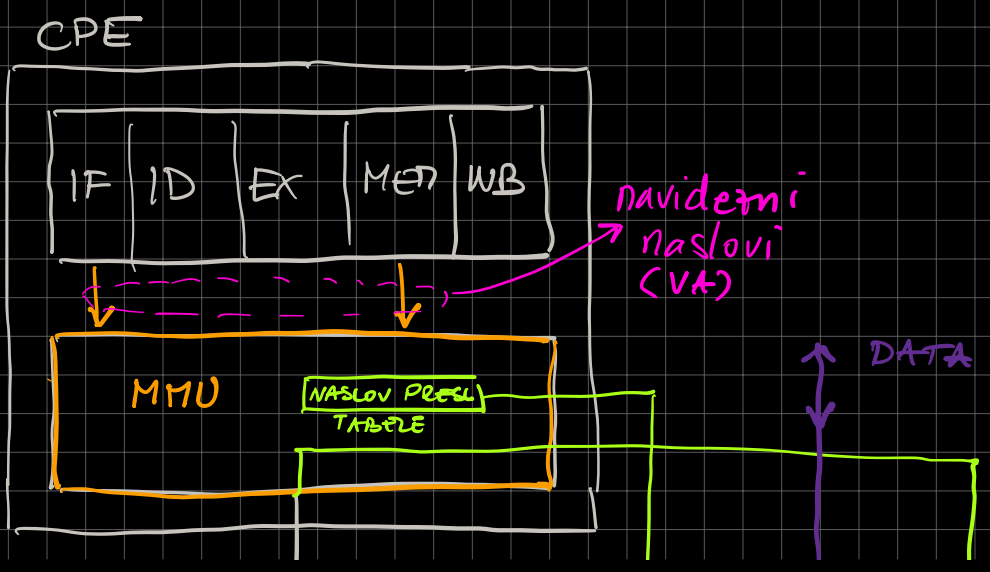
1. da programi in CPE nič ne vedo o fizični RAM pomnilniški
2. da vsi programi mislijo, da si celoten pom. prostor lahko naslavljajo poljubne besede v pom. prostoru in mislijo, da pripadajo le njemu
3. da so programi pozicijsko neodvisni

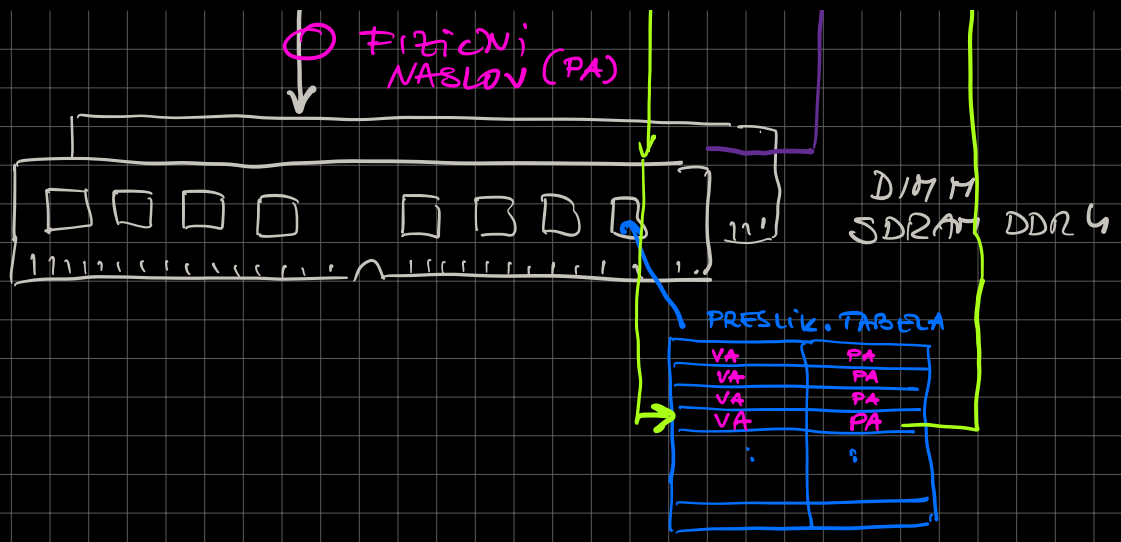
⇓

Rešitev: **VIRTUALNI POMNILNIK**

↙

CPE/programi ne vedo kje in koliko je fizične RAM-a





Kakšna naj bo preslikovalna tabela ?

1. Naj ne vsebuje preširoke za vsak naslov posebej, ker bi bila prevelika (npr. imela bi 2^{64} vnosov)

2. Naj preslikuje naslove blokov zaporednih pom. besed

⇓ ⇓ STRANI (PAGES)

Na ta način bodo pom. besede, ki jo zaporedne v navideznem pomnilniku, zaporedne tudi v fizičnem

⇓

OHRANJAMO LOKALNOST !

⇓

OSTRANJEVANJE (PAGING)

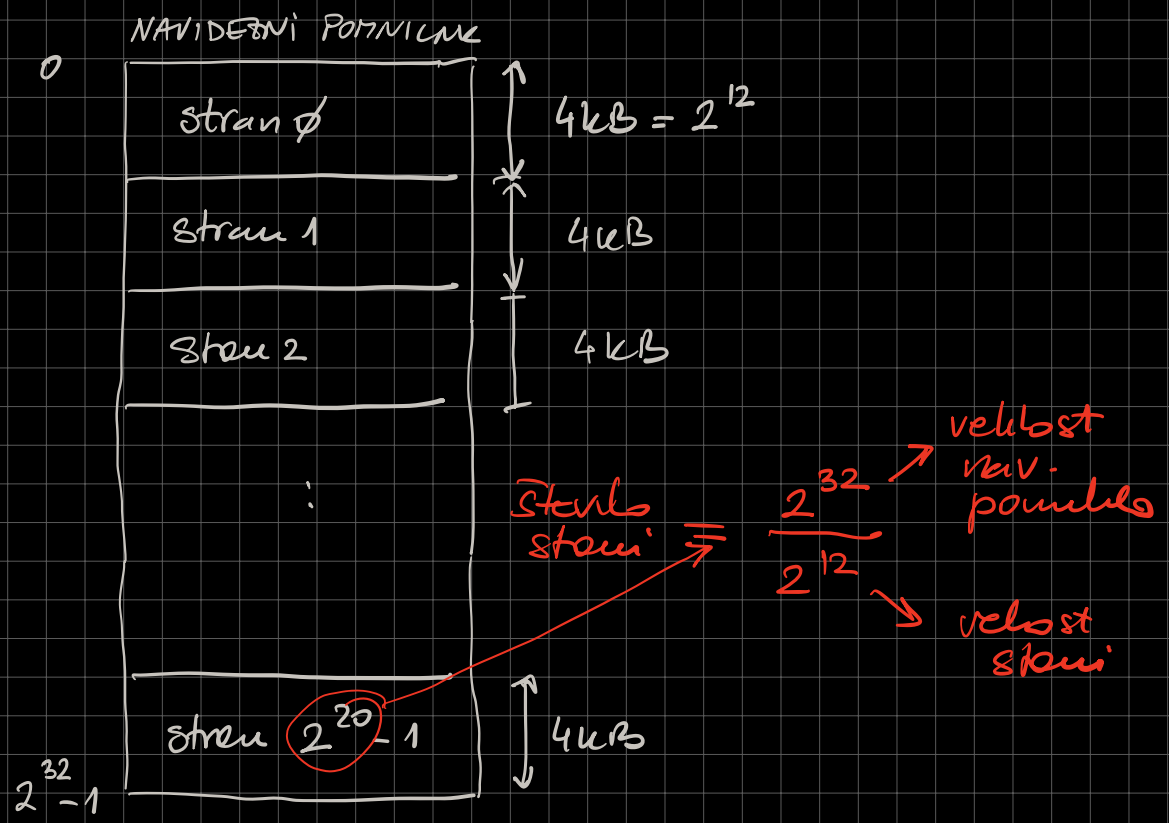
konkreten pogled ostranjevanja

→ dolžina naslova: 32 bitov (n)

→ velikost strani: 4kB

⇓

Navidezni naslovni prostor razdelimo na bloke (strani) velikosti 4kB:



IDEJA: preslikovalna tabela naj vsebuje preslikave le za "vsako prvo peredo posamezne strani"

⇓

Velikost prostora za STRANE (ne za posamezne pom. besede)

⇕

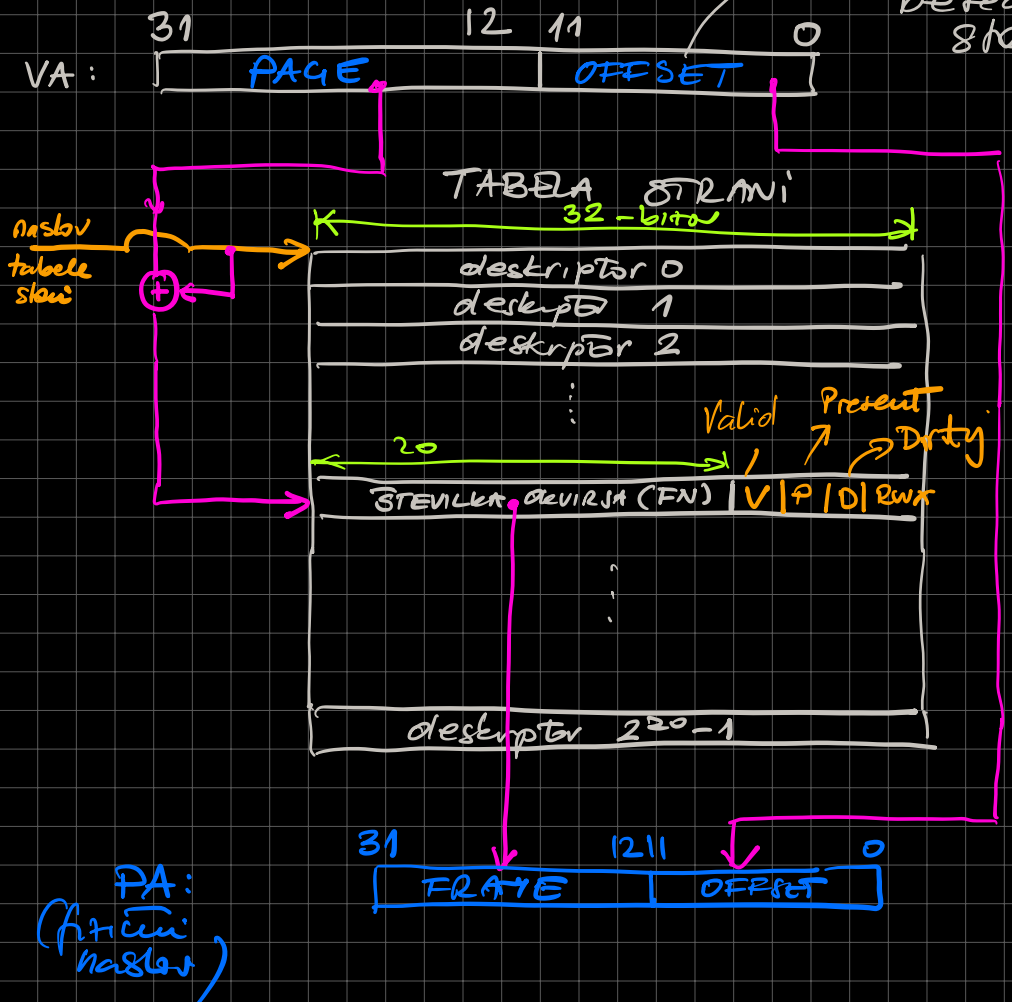
TABELA STRANI (PAGE TABLE)

Tabela strani bo vsebovala 2^{20} vrstic =>
1 vrstica za 1 stran

1 vrstica naj hrani fizični naslov posamezne strani

deskriptor strani

→ odmik pom. besede znotraj strani



1. CPE izstavi VA (32-biten)

2. MMU enota prestreže VA in na osnovi zgornjih 20 bitov ve, kateri deskriptor hrani fizični naslov (= frame number)

3. MMU vzame teh 20 bitov, jih prišteje začetnemu naslovu tabele strani in tako dobi naslov deskriptorja v RAM-u

1. dostop do RAM-a

4. S tem naslovom prebere deskriptor

5. Če je [polje na bitu V, P, D, RWX] vse OK, gre MMU enota s fizičnim naslovom, ki ga dobi tako, da FN bitom doda offset iz VA, v RAM ⇒ to je naslov podatkov, ki jih želi CPE

2. dostop do RAM-a

6. Če je pa karkoli narobe: NAPAKA STRANI (PAGE FAULT)

MMU proži prekinitev ⇒ OS upošteva naslov in ustrezno ukrepa

Večnivojsko ostranjevanje

npr. če je naslov dolg 64 bitov in je stran velika 4kB

⇓

V navid. pomnilnik gre $2^{64}/2^{12} = 2^{52}$ strani

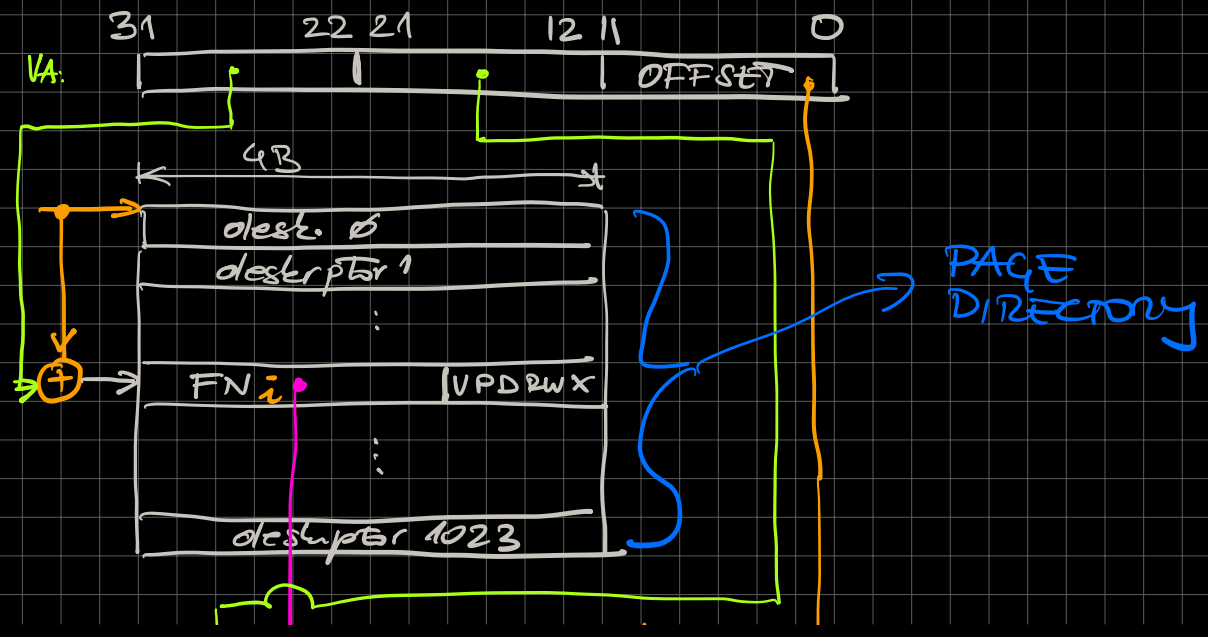
Tabela strani je prevelika, da bi šla v RAM

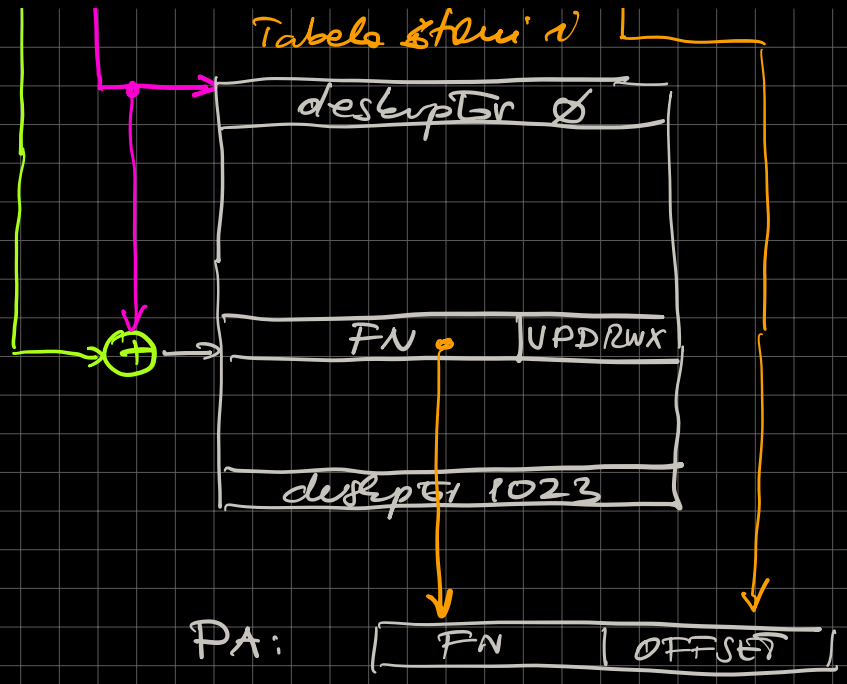
IDEJA: DREVESNA STRUKTURA:

→ Tabelo strani razkosamo v manjše kose ⇒ praviloma velikosti 4 kB ⇒ LISTI DREVESA

→ Dodamo še eno vrhnjo (ROOT) tabelo, ki bo vodila evidenco, kje so ti kosi tabele strani

Zgled IA-32:





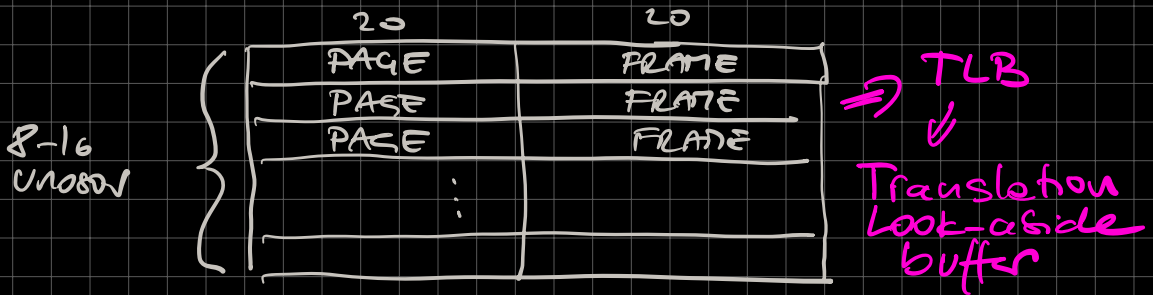
⊕ Zmanjšali količino deskriptorjev, ki jih moramo hraniti v RAM-u

⊖ Če en dodaten dostop do pomnilnika ⇒ Tri(!) dostopi !!!?

Kako pohitriti prevajanje VA → PA?

⇓

V MMU vgradimo en majhen pomnilnik:



⇒ TLB

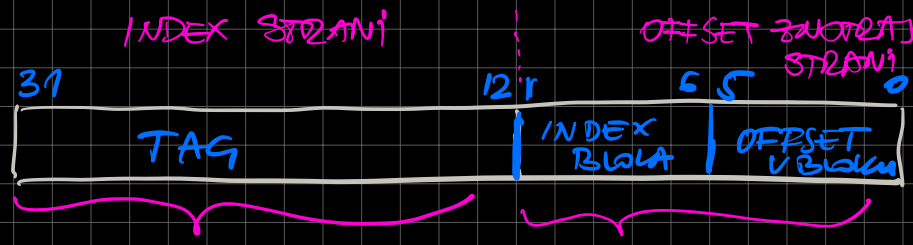
⇓

Translation Look-aside Buffer

Sedaj MMU najprej pogleda ali je iskan stran že kopirana v TLB


če je => BINGO
poiščem FRAME NUMBER iz TLB-ja in je to fizični naslov operanda/ukaza
če ni => INTERRUPT!
OS
pomeni napako
Čakaj, čakaj, kaj pa predpomnilnik?!?
če naslavljam s fizičnim naslovom
ga moram najprej pridobiti iz navideznega
če naslavljam z navideznim naslovom:
več istih navideznih naslovov se preslika v isti blok v PP
SPOMNIMO SE: kaj pomeni naslavljat PP?
1. kje v PP je naslednji naslov
BLOCK/SET
INDEKSIRANJE
2. PRIMERJAMO TAG-a =>
TAGIRANJE

Zgled: - velikost bloka = 64 B
- Število blokov 64
} 4 kB p.p.



TAGIRANJE: moramo preslikati naslov v fizičnega

⇓

vpam in računam, da sta ima v TLB-ju

Ker se ta del naslova ne spremeni, lahko začnemo operacijo s predpomnilnikom

↓ ↘

Istočasno berem PP in TLB
če ujeto TAG-a enako ⇒ HIT
v >99% dostopov primerov

⇓

VIPT predpomnilnik

Virtually Indexed, Phisically Tagged

NI PISNEGA IZPITA

USTNI IZPIT

↳ sledonj vsak dan v tednu

2 x / dan

8:30 — 10:30

17:30 — 19:30

Vi: izberite termin in se prijavite preko STUDISA!

TI ZELO MI BODO TEDENSKO RAZPISANI ✓ STUDISU