14/1/20

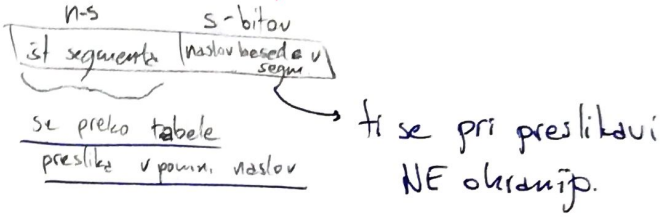
tabela segmentov (segment table) – preko nje je org. preslikovalna funkcija

↓št segm. ni fiksno; velikost tabele ni znana vnaprej

Rešitev: vsak prog. ima SVOJO TABELO SEGM.

Vsak segment ima v tabeli s. svoje polje: **deskriptor segmenta**

n-bitni nov. naslov, ki ga tvori CPE:



① **Prisotni bit P** – ekvivalenten P bitu pri ostranjevanju

↳ 1: seg. je v GP, naslov segm. nam pove KJE (na katerem naslovu)
dostop odvisen od RWX IN če relativni naslov ≤ velikost segmenta L

↳ 2: seg. ni v GP; poskus dostopa – zgrešitev, napaka segmenta

②
↳ Past, PSP prenese segment iz Pom. P v GP
↳ SA (naslov segmenta) (parameter) ni veljaven

② **Zaščitni ključ RWX** ≈ ostr.

↳ zaščita lahko popolnejša kot pri ostr.
↳ lahko poverno, ali se bo seg. spreminjal (velikostno),
ali ga lahko uporablja več uporabnikov
...

③ **Umazani bit C** (change) = = ostr.

↳ ko gre seg. v GP, se nastavi na 0, če se piše v segm, gre na 1.

④ **Velikost segmenta L** (length) → ni fiksna velikost seg.

↳ vsebuje trenutno velikost seg. (merjeno v št. besed); max je 2^s besed.

↳ dolg s bitov

↳ pri vsakem dostopu se s-bitni naslov znotraj seg. primerja ≥ L
↳ naslov > L: napaka (past) → detekcija prog. napak in nedovoljenih posegov.

336

14/1/20

(5) Naslov segmenta SA (Segment Address)

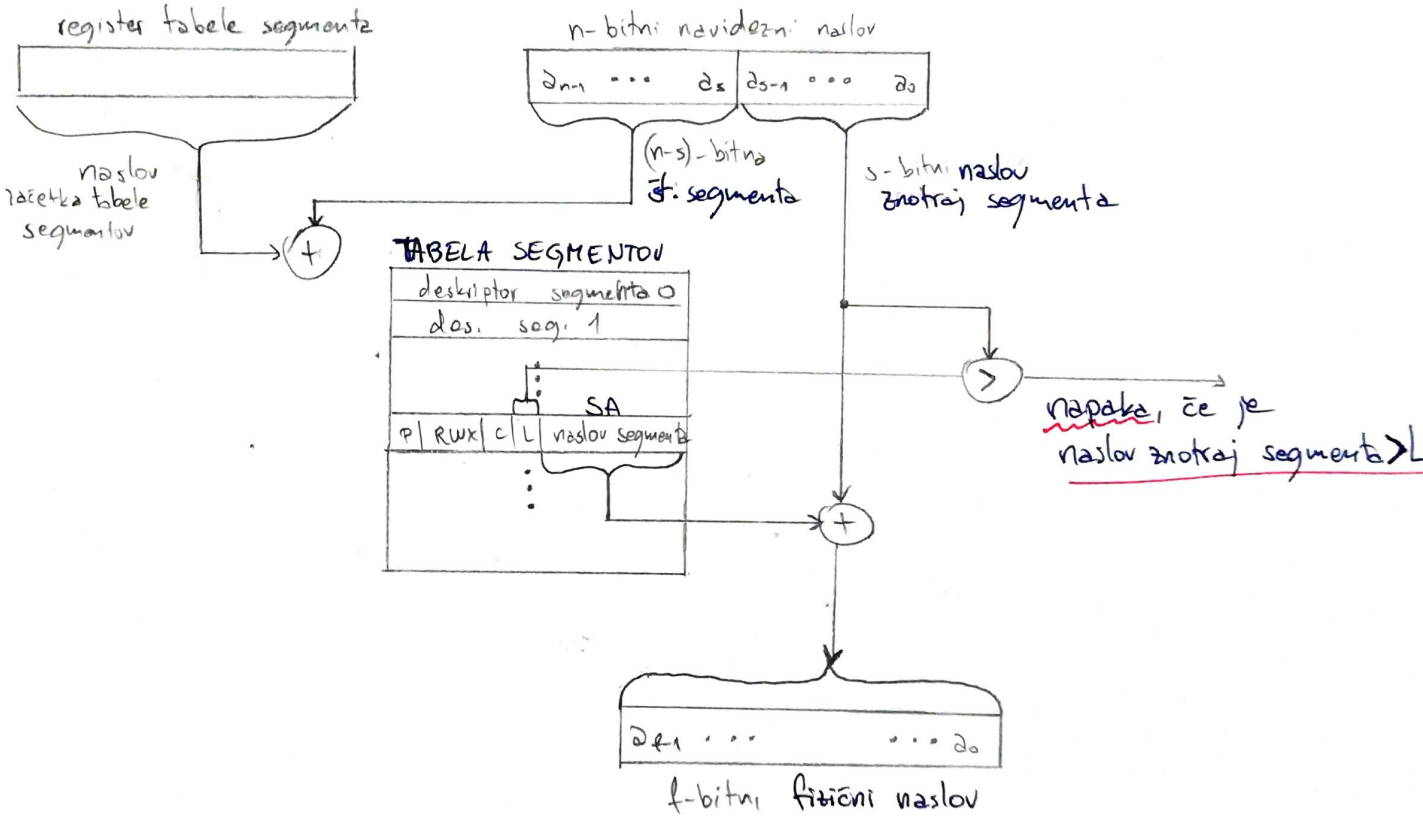
↳ vsebuje fizični naslov v GP, kjer je segment.

↳ do dostopa se temu naslovu prišteje naslov znotraj segmenta

fizični naslov besede

↳ kar je naslov segm. SA poljuben, spodnji naslovni biti fizičnega n. NISO ENAKI bitom NN.

Prostor ni linearen, oz je le znotraj segmenta.



13/1/20

Pri računanju z NN se št. segmenta (zgornjih n-s bitov NN) ne more spremeniti.
Če bi to želeli, se sprožI past.

Obstoječi segmenti - seg. ki se trenutno izvajajo
Tabela obstoječih seg - za vsak obstoječi o vseboje par [ime seg., št. seg.]
↳ simbolično. Pod tem, je znan programerju imenom je s.

2 možnosti delovanja rač s seg.:

① – podoben tistemu rač. ki nima nav. P. (339)

② – dinamično povezovanje (dynamic linking)

↳ v tabelo obstoječih S in v tabelo s. prog. se upiše samo seg., ki vsebuje začetek programa (pri ① se premesejo vri segmenti) pred začetkom

↳ ko ta med izvajanjem prvič klIče nek drug seg., povezovalnik poišče datoteko s tem seg. in mu preko tabele obst. seg (TOS) dodeli št. in nov deskripto v tabeli seg. programa (TSP).

+ povežejo se samo tisti seg., ki jih prog. res uporablja

KAKO LAHKO VEČ PROG. UPORABLJA ISTI SEGMENT?

Razdelitev seg. v 2 vrsti:

① Lokalni segmenti - pripadajo 1 programu (dostop drugim prog NI dovoljen)

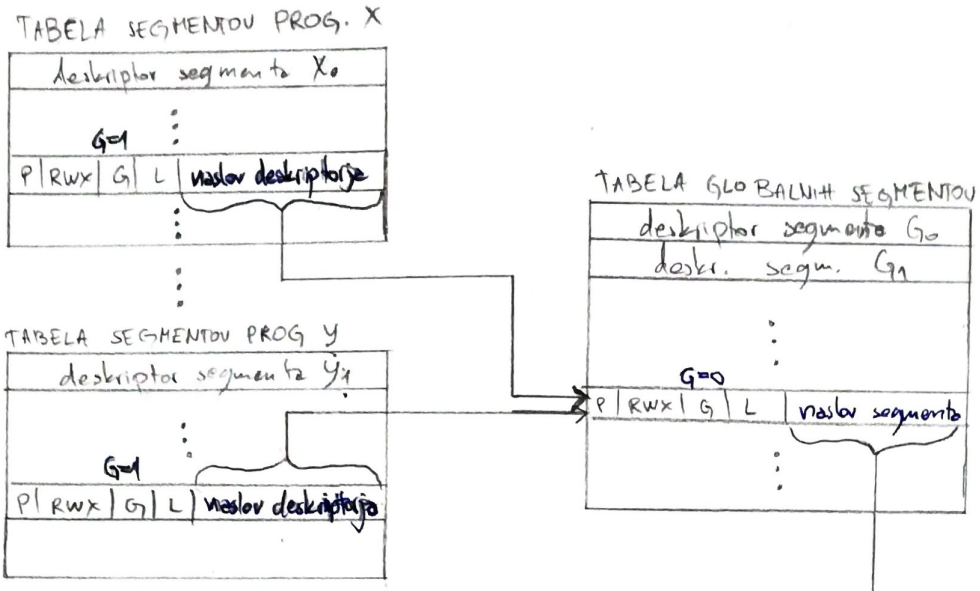
② Globalni segmenti - lahko dostopajo usi prog. Npr. sistemski prog.

↳ deskriptorji vsebujejo še bit G – [1] - globalen. Vsebuje kazalec na deskriptor v tabeli GS (globalnih segmentov)
kot naslov segmenta se uporablja parameter iz GS tabele

– [0] - preslikovanje ostane enako

(338) (slika)

14/1/20



Lahko pride (pri ↓segm.) do prenetavanja.

↳ zaradi različne velikosti seg., je v GP potrebno najti ustrezno velik prostor.

↳ imamo luknje

↳ če so vse luknje premajhne od velike za segment, vsaka posamezna pa ne:

ZUNANJA FRAGMENTACIJA (precejšen problem)

(pri ostranjevanju je ni), (pri segmentaciji ni notranje fragmentacije)

obe rešitvi glede na izkoriščenost enako škodljivi

vendar pri notranji vsaj vemo max in avg. velikost.

Zunanja se stalno spreminja ☹

Algoritmi za zunanjo fragmentacijo: → imamo k lukenj, označimo jih $S_1, S_2, \dots, S_k$

S_i ... velikost luknje i

$S_1 \leq S_2 \leq \dots \leq S_k$

S ... segment velikosti S, ki ga želimo prenesti

① Najboljše ujemanje: poiščemo min i, da bo $S \leq S_i$

↳ segment prenesemo v luknjo S_i. Razlika najmanjša

⊖ rezultat so zelo majhne luknje, ki skoraj zanerljivo ne bodo uporabne

② Najslabše ujemanje: Luknje razvrstimo po njihovi padajoči vrednosti.

$S_1 \geq S_2 \geq \dots \geq S_k$. Velja: $S \leq S_i$ (isto kot pri ①). Razlika je največja.

⊖ Luknje so večje, vendar jih hitro zmanjka (i je min.)

③ Prvo ujemanje: Luknje razvrščene po naraščajočih začetnih naslovih: ne glede na velikost

Luknja z min indeksom i, ko je $S \leq S_i$, je tista, v katero gre segment.

Razlika naključna.

⊕ najboljši performance na meritvah

16/1/20

Strnjevanje pomnilnika - počasna operacija

↳ izvede se, ko uporaba pomn. zaradi fragmentacije ni več možno

↳ postopek, ko vse luknje strnemo v prazen prostor

↳ vmes izvajanje programov stoji

☹ ↳ slabo. Čista oblika segmentacije se uporablja redko

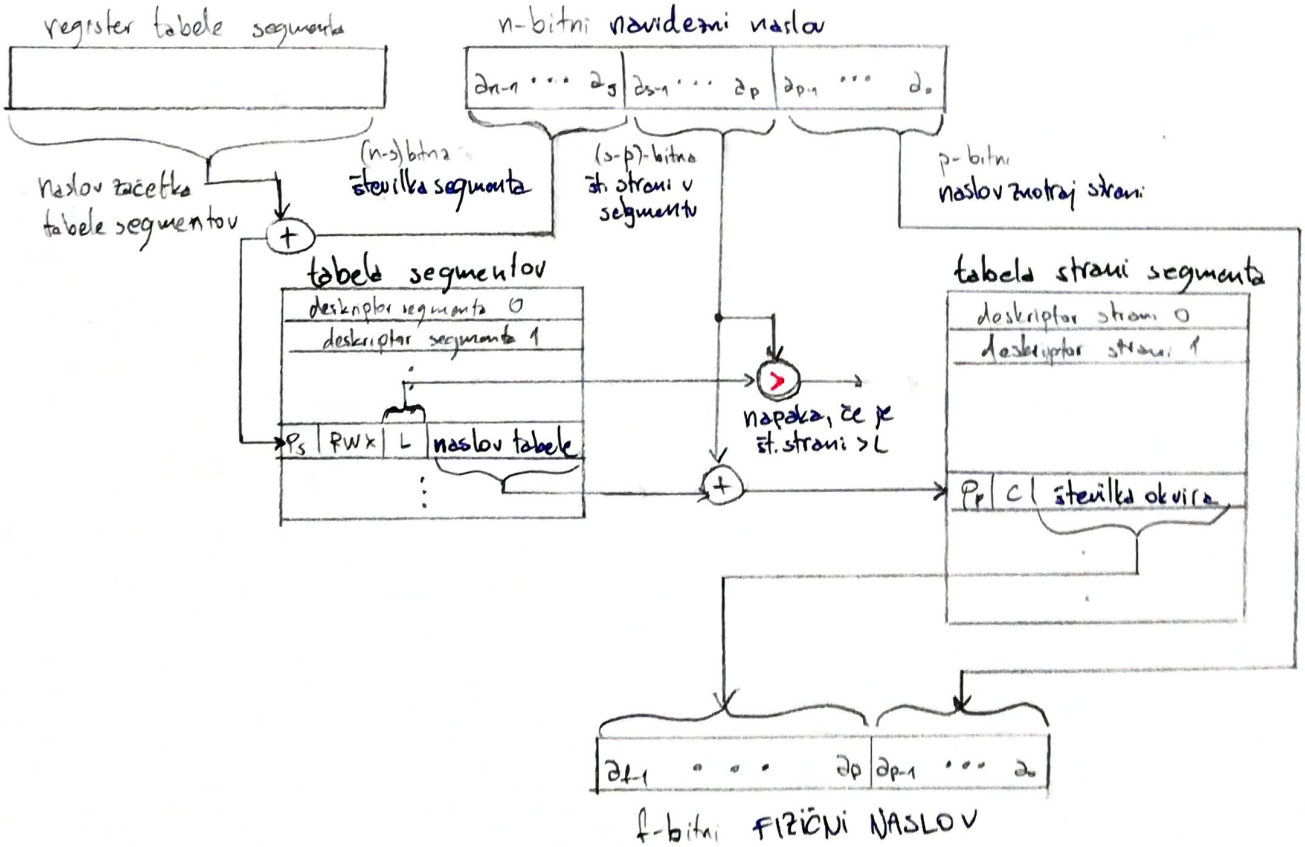
(339) 9.3.3 segmentacija z ostranjevanjem

Vsak segment razdelimo na strani z načinom ostranjevanja.

To je: linearna segmentacija (linear segmentation)

Vsak prog. ima poleg TS (tabela segmentov) tudi množico tabel strani → po 1 za vsak segment.

GP razdeljen na okvire strani



14/1/20

Zadnja 2 dela naslova sta v bistvu naslov besede v segmentu, ki je podan v 2 delih.

① Prisotni segment bit P_s (Present)

↳ [1] : tabela strani tega segm. je v GP. PTA pove na katerem naslovu.

↳ [0] : TS (tabela strani) ni v GP. Poskus dostopa do besede v takem segmentu: napaka segmenta
PSP prenese tabelo strani tega segmenta iz Pom. P. v GP.

Dostop odvisen od RWX in če št. strani ni večja od segmenta L (meri se v št. strani)

② Zaščitni ključ RWX == čista segm.

③ Velikost segmenta L - trenutna velikost segmenta merjena v št. strani

↳ max.: 2^{s-p} strani in je določena vnaprej. L mora biti dolg s-p bitov

↳ če je št strani (s-p bitov določa to št.) > L → napaka (detekcija prog. napak in nedovoljenih dostopov)

↳ velikost L se spreminja

④ Naslov tabele strani PTA (Page table address)

↳ fiz. naslov v GP na katerem je tabela strani za ta segment (če je $P_s = 1$)

↳ ob dostopu se PTA prišteje št. strani (v segmentu) $\xrightarrow{SUM}$ fizični naslov deskriptorja v tabeli strani

Ko je $P_s = 1$, se preslikovanje nadaljuje preko tabele strani.
Naslov te tabele je glavni rezultat, ki ga opravi tabela segmentov.

① Prisotna stran bit P_s (present) → ni potrebno, da bo bil segm. vedno v celoti v GP. (Drugače kot pri čisti segmentaciji)

↳ [1]: stran je v enem izmed okvirov v GP. Fiz N. (iskani) = št. okvirja + spodnjih p bitov NN.

↳ [0]: strani ni v GP: napaka strani → PSP poseruisiva in prenese stran iz Pom.P v GP

② Umazani bit C == ostranjevanje.
↳ nekateri sistemi imajo bit sprememba tudi v tabeli segmentov.

③ Številka okvirja FN (Frame number) == ostranjevanje

seg. z ostr. IZBOLJŠAVE:

⊕ odstranitev zunanje fragmentacije. Ni več zamudnega stranjevanja pomn.

⊕ boljša izkoriščenost prostora, ki ga zasedajo segmenti. V GP ne rabijo biti celi segmenti.
Lahko so le tisti, ki se trenutno uporabljajo.
(dovolj jel)

15/1/20

9.4 PROBLEMI PRI REALIZACIJI N. P.

stanje programa - tabela strani in segmentov + PC + registri

programski števec