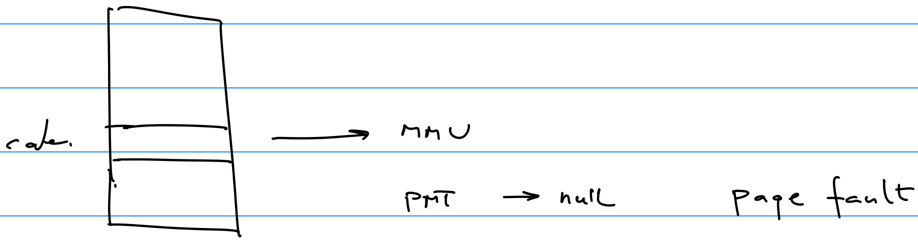
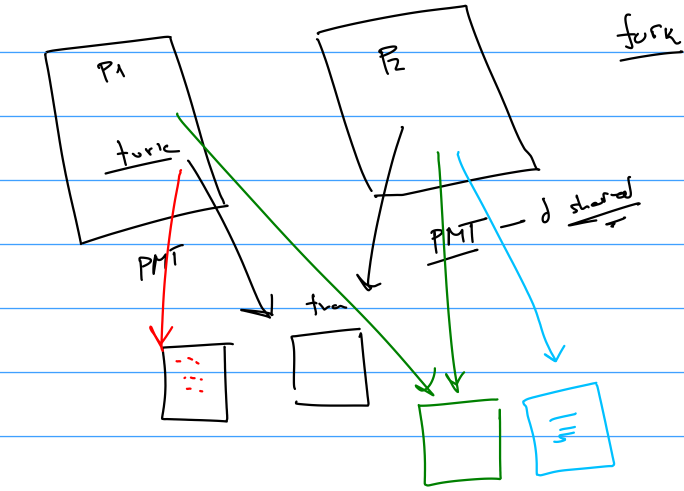
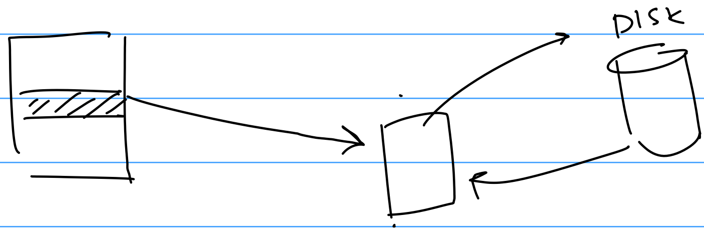
demand paging — загрузка страницы по запросу.



copy-on-write



page replacement — замена страницы.



3. (10 poena)

Neki sistem sa straničnom organizacijom memorije, sa učitavanjem stranica na zahtev (*demand paging*), ali bez zamene stranica (*page replacement*), ima veličinu stranice od 4 KB i koristi tehniku kopiranja pri upisu (*copy on write*). Neki roditeljski proces ima jedan logički segment za programski kod, veličine 560 KB, jedan logički segment za statičke podatke, veličine 1076 KB, i jedan logički segment za stek veličine 1 MB. Do trenutka pokretanja procesa deteta sistemskim pozivom *fork*, ovaj roditeljski proces je pristupao svim stranicama navedenih logičkih segmenata za kod i statičke podatke, a prilikom tog poziva stek mu je veličine 276 KB. Nakon uspešnog sistemskog poziva *fork*, povratkom iz poziva potprograma, proces roditelj najpre smanjuje veličinu svog steka do minimalne veličine od 128 KB, da bi je do kraja svog izvršavanja povećavao i do maksimalnih 560 KB; veličina steka ne izlazi iz tog opsega. Tokom tog perioda svog izvršavanja, on menja statičke podatke koji ukupno zauzimaju 16 stranica. Slično, proces dete, nakon dobijanja kontrole povratkom iz sistemskog poziva *fork*, najpre smanjuje svoj stek do minimalnih 200 KB, a potom povećava do maksimalnih 640 KB; tokom tog perioda svog izvršavanja, on menja statičke podatke koji ukupno zauzimaju osam stranica, od kojih se dve poklapaju sa stranicama koje je nakon poziva *fork* menjao roditelj.

Izračunati koliko maksimalno fizičke memorije zauzimaju ova dva procesa tokom celokupnog svog izvršavanja i obrazložiti postupak. Rezultat izraziti u broju okvira i u KB. Navesti i koliko fizčikih okvira zauzimaju ukupno njihovi segementi za kod, statičke podatke i stekove.

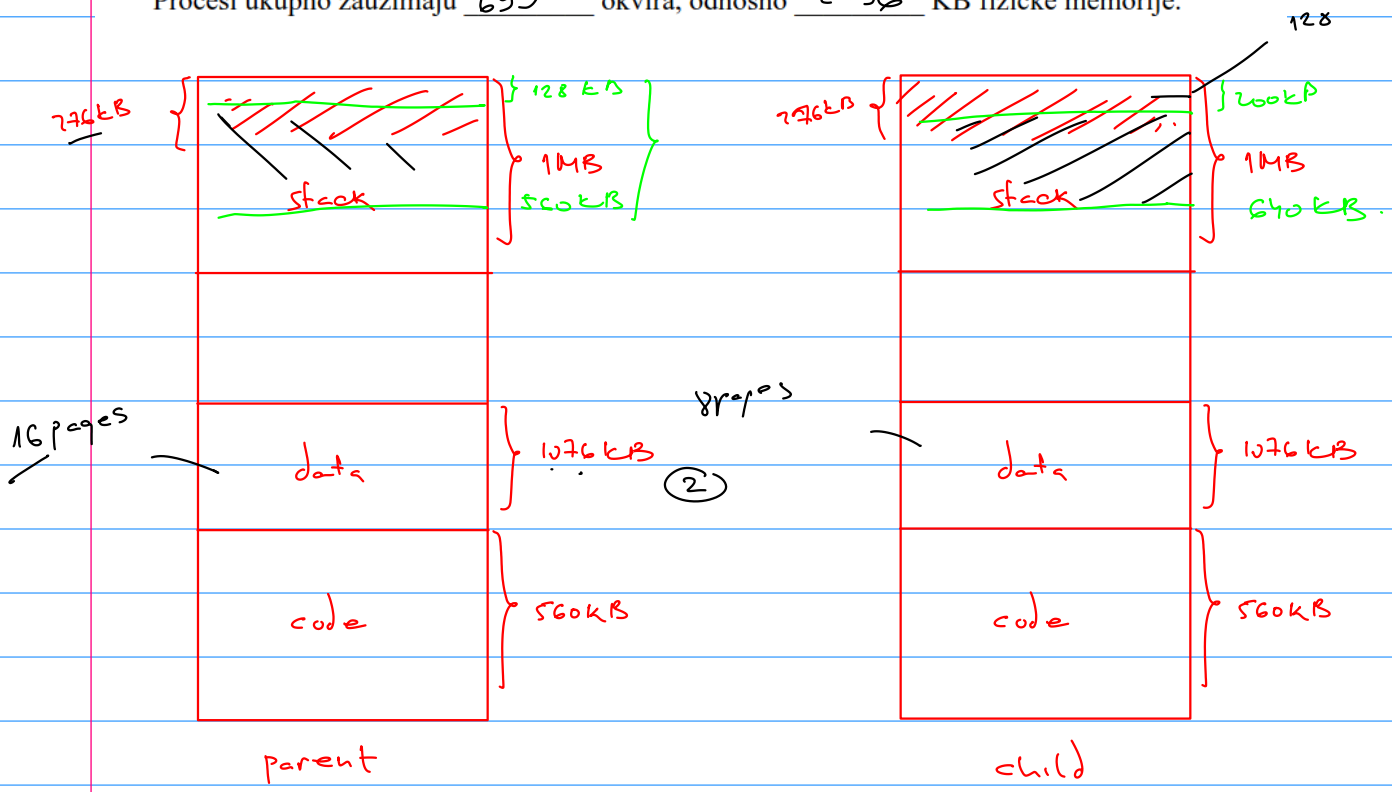
Odgovori:

Segmenti za kod ukupno zauzimaju 140 okvira.

Segmenti za statičke podatke ukupno zauzimaju 291 okvira.

Segmenti za stekove ukupno zauzimaju 268 okvira.

Procesi ukupno zauzimaju 699 okvira, odnosno 2796 KB fizičke memorije.



$\frac{560\,kB}{4\,kB} = 140$, $\frac{1076\,kB}{4kB} = 269$, $\frac{1MB}{4kB} = \frac{2^{10}\,kB}{4\,kB} = 2^8 = 256$

$\frac{276\,kB}{4kB} = 69$.

$\frac{560 - 128}{4} = 108$, $\frac{640 - 128}{4} = 128$

$\frac{128\,kB}{4kB} = 32$

$32 + 108 + 128 = 268$

3. (10 poena)

Neki sistem sa straničnom organizacijom memorije koristi tehniku kopiranja pri upisu (*copy on write*). Deskriptor stranice u PMT je veličine 32 bita, s tim da najnižih 16 bita sadrži broj okvira, a najviša tri bita koduju prava pristupa *RWX* tim redom (bit *R* je najviši bit).

Za evidenciju svih okvira raspoložive operativne memorije sistem koristi niz `frames`; *i*-ti element ovog niza odgovara okviru broj *i*. Vrednost ovog elementa označava broj stranica koje dele isti okvir pre razdvajanja pri upisu; ako je ta vrednost 0, okvir je slobodan; ako je ta vrednost 1, okvir koristi samo jedna stranica jednog procesa; ako je ta vrednost veća od 1, taj okvir dele stranice različitih procesa.

Kada obrađuje hardverski izuzetak zbog pokušaja nedozvoljenog upisa, nakon provere da adresirana stranica pripada segmentu koji je logički dozvoljen za upis i da treba raditi kopiranje pri upisu, jezgro poziva operaciju `copyOnWrite` sa argumentom koji pokazuje na deskriptor stranice u koju je pokušan upis. Ta funkcija tada treba da iskopira sadržaj okvira u nov okvir i da stranicu koja je referencirana preusmeri u taj okvir i dozvoli upis u tu stranicu. Ostale stranice koje su koristile isti okvir, čak i ako je to samo jedna preostala, se ne menjaju. Kada se pokuša upis u poslednju preostalu stranicu koja je delila taj okvir, ova funkcija će samo dozvoliti upis u nju (nema potrebe kopirati je, jer je jedina).

Implementirati funkciju: `int copyOnWrite (uint32* pd)`. Na raspolaganju je sledeće:

- `getFreeFrame()`: vraća broj prvog slobodnog okvira; ukoliko takvog nema, vraća 0;
- `copyFrame(uint16 oldFrame, uint16 newFrame)`: kopira sadržaj iz okvira broj `oldFrame` u okvir broj `newFrame`.

Ako slobodnog okvira nema, funkcija `copyOnWrite` ne treba ništa da menja, samo da vrati status -1. U slučaju uspeha, ona treba da vrati status 0. Tipovi `uint16` i `uint32` predstavljaju neoznačene 16-bitne, odnosno 32-bitne cele brojeve, respektivno.

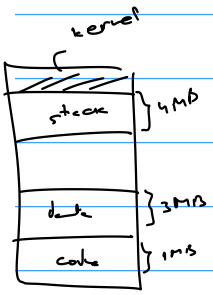
```
int copyOnWrite (uint32 *pd) {
    uint16 frame = (uint16)((*pd) << 3) >> 3);
    if (frames[frame] > 1) {
        *pd |= ((uint32) 1) << 30;
    } else {
        uint16 new_frame = getFreeFrame();
        if (new_frame == 0) return -1;
        copyFrame(frame, new_frame);
        *pd >>= 16;
        *pd <<= 16;
        *pd |= (uint32) new_frame;
        *pd |= ((uint32) 1) << 30;
    }
    return 0;
}
```

3. (10 poena)

Virtuelni adresni prostor nekog računara je 4GB i organizovan je stranično, adresibilna jedinica je bajt, a PMT je organizovana u dva nivoa. Broj ulaza u PMT oba nivoa je isti, kao i širina svakog ulaza koja je po 32 bita. PMT svakog nivoa zauzima tačno jednu celu stranicu.

Neki operativni sistem na ovom računaru zauzima najnižih 4MB raspoložive fizičke memorije za potrebe kernela. Kako ne bi menjao memorijski kontekst prilikom obrade sistemskih poziva, sistem preslikava (mapira) najviše stranice svakog kreiranog procesa u ovaj memorijski prostor kernela, s tim što postavlja indikatore u deskriptoru tih stranica u PMT tako da one nisu dostupne u neprivilegovanom (korisničkom) režimu rada procesora.

Kernel koristi tehniku *copy-on-write*. U implementaciji sistemskog poziva *fork*, sve PMT procesa roditelja i deteta, uključujući i PMT prvog nivoa, se ne kopiraju, nego se dele, sve dok je to moguće, odnosno dok kernel ne mora da ih razdvoji (kopira). Osim toga, pošto je deo memorije koji pripada kernelu deljen u virtuelnim adresnim prostorima svih procesa na istom mestu, one PMT koje se odnose na taj deo memorije svi procesi dele sve vreme.



a)(2) Kolika je veličina stranice i koliko ulaza ima PMT prvog nivoa? Prikazati postupak kojim se došlo do odgovora.

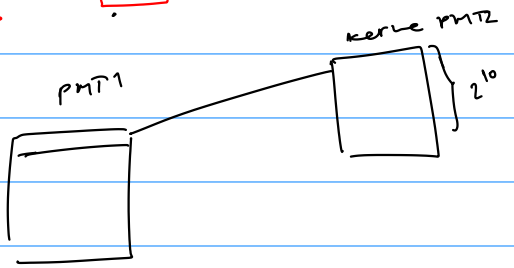
b)(3) Neki proces izvršio je sistemski poziv *exec* koji inicijalizuje memorijsku mapu tog procesa tako da, osim dela koji se preslikava u memoriju kernela, alocira jedan segment za kod veličine 1MB na početku svog virtuelnog adresnog prostora, jedan segment za podatke veličine 3MB odmah nakon segmenta koda, i segment za stek veličine 4MB odmah ispod memorijskog dela koji pripada kernelu. Koliko operativne memorije (osim one koja je prethodno već bila zauzeta zbog postojanja drugih procesa) je alocirano prilikom izvršavanja ovog sistemskog poziva za potrebe smeštanja PMT ovog procesa? Obrazložiti.

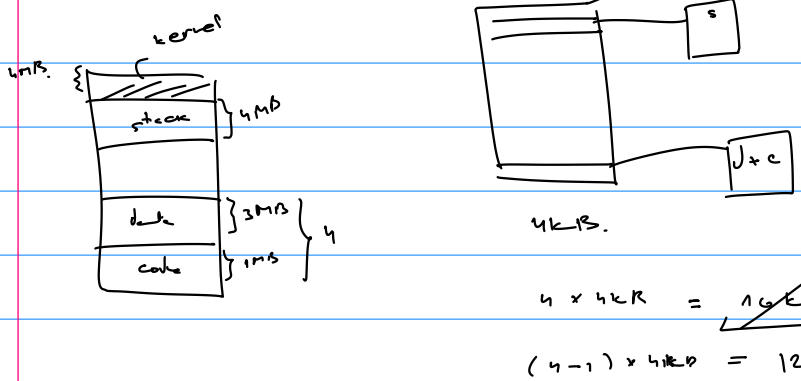
c)(2) Ovaj proces sada izvršava sistemski poziv *fork*. Koliko dodatne memorije (osim one koja je već bila zauzeta zbog postojanja ovog roditeljskog i drugih procesa) je dodatno alocirano za potrebe smeštanja PMT novog procesa deteta prilikom izvršavanja ovog sistemskog poziva? Obrazložiti.

d)(3) Posmatrani proces (roditelj) je potom odmah izvršio instrukciju poziva potprograma, što je uzrokovalo upisom u samo jednu stranicu na vrhu steka. Koliko dodatne memorije (osim one koja je već bila zauzeta pre toga) je dodatno alocirano za potrebe smeštanja PMT oba procesa, i roditelja i deteta zbog izvršavanja ove instrukcije? Obrazložiti.

Rešenje:

$\frac{4MB}{4KB} = 2^{10}$ PMT2 tabela kernel de





$4 \times 4KB = 16KB$

$(4-1) \times 4KB = 12KB.$

10 20 30
K M G

3. (10 poena)

Virtuelni adresni prostor nekog računara je 4GB i organizovan je stranično, adresibilna jedinica je bajt, a PMT je organizovana u dva nivoa. Broj ulaza u PMT oba nivoa je isti, kao i širina svakog ulaza koja je po 32 bita. PMT svakog nivoa zauzima tačno jednu celu stranicu.

VAS = 4 GB

PAGE = n · 32 b = n · 4 B

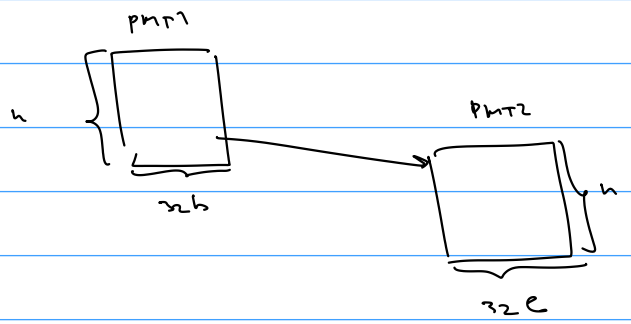
PAGE = 4 KB

$n^2 \cdot PAGE = VAS$

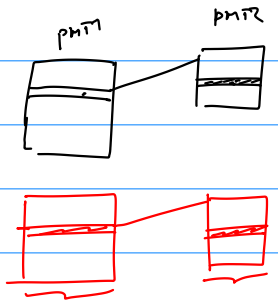
$n^2 \cdot n \cdot 4KB = 4\,GB$

$n^3 = 2^{30} \rightarrow n = 2^{10}$

PMT1 , PMT2



parent child



(1+1) 4KB = 8KB

10 20 30
K M G

page number.

VAS = 4 GB = $4 \cdot 2^{30}$ B = 2^{32} B

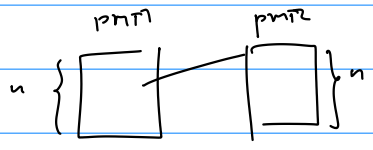
PAGE = 16KB = $2^4 \cdot 2^{10}$ B = 2^{14} B

PAS = 16MB = $2^4 \cdot 2^{20}$ B = 2^{24} B

VA(32) = PAGE1(9) : PAGE2(9) : OFF(14)

OFF(14)

PA(24) = FRAME(10) : OFF(14)



$2^x = n$, $2^y = n \rightarrow x = y$

$x + y = 32 - 14 = 18$

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 4GB i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 16KB. Fizički adresni prostor je veličine 16MB. PMT (*page map table*) je organizovana u dva nivoa, s tim da je broj ulaza u PMT prvog i drugog nivoa isti.

Za svaki proces operativni sistem kreira memorijski kontekst kao skup deskriptora logičkih memorijskih segmenata. Deskriptor segmenta predstavljen je strukturom `SegDesc` u kojoj, između ostalog, postoje sledeća polja:

- `unsigned long startingPage`: početna stranica segmenta u virtuelnom prostoru;
- `unsigned long size`: veličina segmenta izražena u broju stranica;
- `SegDesc *next, *prev`: sledeći i prethodni deskriptor segmenta u dvostruko ulančanoj listi segmenata istog procesa; na prvi segment u listi ukazuje polje `segDesc` u PCB procesa;
- `short rwx`: u tri najniža bita definisana su prava pristupa stranicama datog segmenta, odnosno dozvoljene operacije za taj logički segment.

Kada stranica nije u memoriji ili ne pripada alociranom segmentu, u odgovarajućem ulazu u PMT nalazi se vrednost 0 koja hardveru indikuje da preslikavanje nije moguće; procesor tada generiše izuzetak tipa stranične greške (*page fault*). U slučaju povrede prava pristupa stranici, odnosno nedozvoljene operacije, procesor generiše izuzetak koji naziva *operation denied*. Sistem primenjuje *copy on write* tehniku.

a)(3) Prikazati logičku strukturu virtuelne i fizičke adrese i označiti veličinu svakog polja.

Odgovor:

b)(7) Implementirati funkciju `resolveVAddrExc`:

```
enum AddrExcKind {pageFault, opDenied};

int resolveVAddrExc (PCB* pcb, unsigned long vaddr, AddrExcKind kind,
                     short rwx, SegDesc** ret);
```

Ovu funkciju poziva kod kernela kada obrađuje neki izuzetak prilikom adresiranja virtuelne adrese za proces na čiji PCB ukazuje prvi argument, prilikom pristupa adresi datoj drugim argumentom. Tip izuzetka daje argument `kind`, a operaciju koja je pokušana argument `rwx`.

Ukoliko data virtuelna adresa nije regularna (ne pripada definisanom logičkom segmentu), ili tražena operacija nije dozvoljena za tu adresu, ova funkcija treba da vrati rezultat `MEM_ACCESS_FAULT`, na osnovu koga će kernel ugasiti proces. U suprotnom, radi se o slučaju kada je adresiranje dozvoljeno, ali stranica nije u memoriji i treba je učitati, i kada funkcija treba da vrati `LOAD_PAGE`, ili o slučaju kada treba iskopirati stranicu prilikom upisa, i kada funkcija treba da vrati `COPY_ON_WRITE`. Kad god je virtuelna adresa unutar definisanog logičkog segmenta, u izlazni argument na koga ukazuje argument `ret` treba upisati pokazivač na deskriptor segmenta kom pripada data (regularna) virtuelna adresa.

```
int resolveVAddrExc ( PCB *pcb , ulong vaddr , AddrExcKind kind , ulong rwx , SegDesc **ret){
    ulong page = vaddr >> 14;
    SegDesc *curr = pcb -> segDesc;
    while (curr ≠ null) {
        if ( page ≥ curr -> startingPage && page < curr->startingPage+curr->size) {
            break;
        }
        curr = curr -> next;
    }
    *ret = curr;
    if (curr == null) {
        return MEM_ACCESS_FAULT;
    }
```

```
if (kind == opDenied) {
    if (cm->rwx & rwx) {
        return COPY_ON_WRITE;
    } else {
        return MEM_ACCESS_FAULT;
    }
} else {
    return LOAD_PAGE;
}
}
```

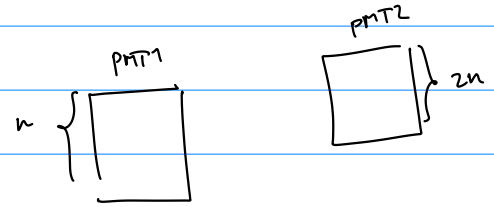
10 20 30
K M G

VAS = 4GB = 2^{32} B

PAGE = 8KB = 2^{13} B

VA(32) = PAGE1(9) : PAGE2(10) : OFF(13)

OFF(13)



$[x + y = 32 - 13 = 19]$

$2^x = n \quad 2^y = 2n$

$\frac{2^y}{2^x} = \frac{2n}{n} = 2$

$2^{y-x} = 2 \longrightarrow y - x = 1$

$2y = 10 \longrightarrow y = 10$

$x = 9$

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 4GB i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 8KB. PMT (*page map table*) je organizovana u dva nivoa, s tim da je broj ulaza u PMT prvog nivoa dva puta manji od broja ulaza u PMT drugog nivoa.

Posmatra se jedan proces kreiran nad sledećim programom:

```
#define N 0x1000
int src[N], dst[N];

int main () {
  for (int i=0; i<N; i++) dst[i] = src[i];
}
```

pod sledećim pretpostavkama:

- tip `int` je veličine 32 bita; promenljivu `i` je prevodilac fomirao kao registarsku promenljivu (njena vrednost se ne čuva u operativnoj memoriji);
- segment za kod ovog programa veličine je jedne stranice, a segment za stek je veličine 32 stranice;
- nizovi `src` i `dst` smešteni su jedan odmah iza drugog u segment memorije alociran za statičke podatke; ovaj segment je onoliki koliko je najmanje stranica potrebno za smeštanje ovih nizova;
- operativni sistem ne učitava nijednu stranicu pri kreiranju procesa, već sve stranice učitava tek na zahtev (*demand paging*).

a)(3) Prikazati logičku strukturu virtuelne i označiti veličinu svakog polja.

Odgovor:

b)(4) Koliko straničnih grešaka (*page fault*) će izazvati izvršavanje ovog procesa ako je operativni sistem za ovaj proces odvojio dovoljno okvira operativne memorije da smesti sve stranice koje taj proces adresira? Obrazložiti.

Rešenje:

c)(3) Ako je operativni sistem za ovaj proces odvojio samo 4 okvira operativne memorije, a za zamenu (izbacivanje) bira onu stranicu istog tog procesa koja je najdavnije učitana u memoriju, koja stranica će biti prva izbačena iz memorije i kojom stranicom će biti zamenjena (stranice imenovati kao *n*-te stranice segmenta za kod, stek ili podatke)? Obrazložiti.