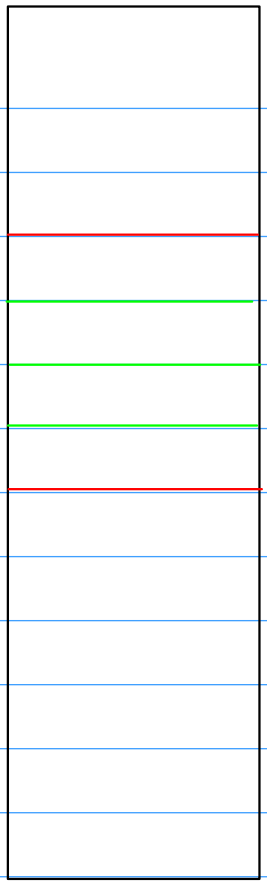


VA ≡ SEG : PAGE : OFF

SMT PMTP

SMTP MMU → PA

10 20 30

k M G

2. (10 poena)

Neki računar podržava segmentno-straničnu organizaciju virtuelne memorije, pri čemu je virtuelni adresni prostor veličine 8GB, adresibilna jedinica je 16-bitna reč, a fizički adresni prostor je veličine 512MB. Maksimalna veličina segmenta je 32MB, a stranica je veličine 512B.

a)(5) Prikazati logičku strukturu virtuelne i fizičke adrese i označiti širinu svakog polja.

AU = 16b = 2B

8GB = 4GAU = 2^{32} AU

512MB = 256MAU = 2^{28} AU

$\frac{8GB}{32MB} = \frac{2^{33}B}{2^{25}B} = 2^{33-25} = 2^{8}$

$\frac{32MB}{512B} = \frac{2^{25}B}{2^{9}B} = 2^{16}$

VA(32)

PA(28)

SEG(8)

PAGE(16)

OFF(8)

512B = 256AU = 2^{8}AU

b)(5) Napisati heksadecimalni kod fizičke adrese u koju se preslikava virtuelna adresa u segmentu 45h, stranici broj 23h u tom segementu, reč broj FFh u toj stranici, ako se ta stranica preslikava u okvir broj FF00h.

frame

VA ≡ SEG : PAGE : OFF

32 = 8 + 16 + 8 ✓

VA(32) ≡ SEG(8) : PAGE(16) : OFF(8)

PA(28) ≡ FRAME(20) : OFF(8)

5 - 00...0

45h : 23h : FFh.

FF00h : FFh. {0X0FF0FF}

K M G T ...
10 20 30 40

PMT

VA ≡ PAGE : OFF

FRAME : RWX V

PA (64)

64 VA(64) $2^{64}B$ FRAME(52)

4KB

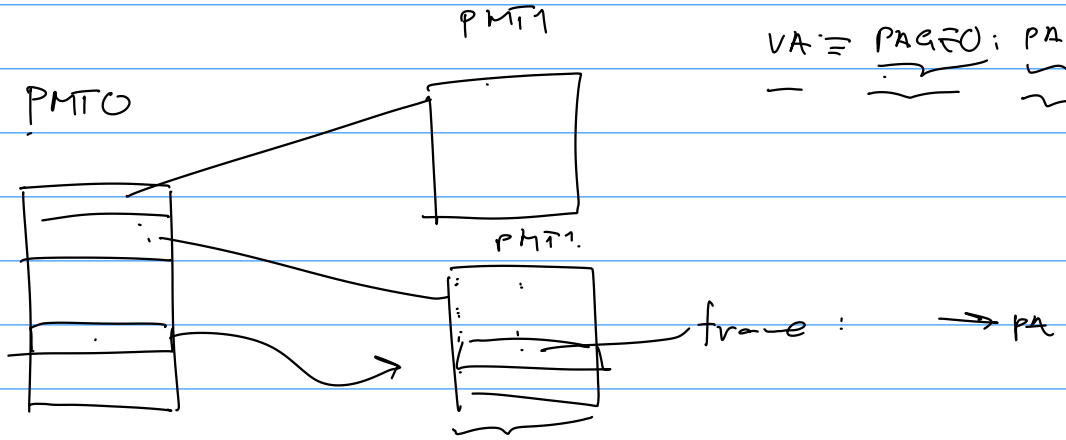
$\frac{2^{64}B}{2^{12}B} = 2^{52}$

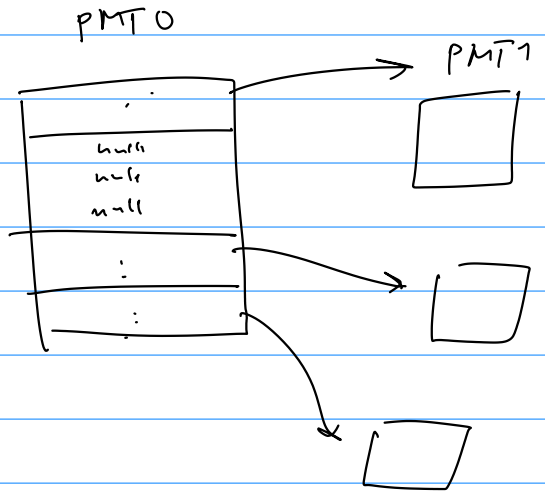
64B
8B.

$2^{52} \cdot 8B = 2^{55}B = 2^{15}TB$

SEG : PAGE

VA ≡ PAGE0 : PAGE1 : OFF





3. (10 poena)

Neki sistem koristi straničnu organizaciju memorije sa stranicom veličine 128 KB. Adresibilna jedinica je 16-bitna reč, virtuelni adresni prostor je veličine 8 GB, a fizički adresni prostor je veličine 2 GB. PMT je u jednom nivou. U deskriptoru stranice u PMT najviša dva bita koduju prava pristupa, dok je u najnižim bitima broj okvira; ako stranica nije alocirana, ceo deskriptor ima vrednost 0.

a)(3) Prikazati logičku strukturu virtuelne i fizičke adrese i označiti širinu svih polja.

b)(7) Implementirati funkciju `sharePages` koja se koristi u implementaciji sistemskog poziva kojim se zahteva logičko deljenje niza od `cnt` susednih stranica počev od stranice `pg1` procesa čija je PMT data parametrom `pmt1` sa nizom od `cnt` susednih stranica počev od stranice `pg2` procesa čija je PMT data parametrom `pmt2`, pri čemu su pomenute stranice drugog procesa već alocirane u okvire, a pomenute stranice prvog procesa nisu, već treba da se dele sa drugim. Ukoliko drugi proces nije alocirao sve pomenute svoje stranice ili je prvi proces već alocirao neku od svojih pomenutih stranica, ova funkcija treba da vrati grešku (negativnu vrednost). Tip `long` je širine 64 bita, tip `int` 32 bita, a tip `short` 16 bita.

```
typedef unsigned short ushort;
typedef ushort PMT[PMT_SIZE];
int sharePages (PMT pmt1, ushort pg1, PMT pmt2, ushort pg2, ushort cnt);
```

pg1, pg1+1, ..., pg1 + (cnt-1) pmt1.

pg2, pg2+1, ..., pg2 + (cnt-1) pmt2.

PMT je ushort[PMT_SIZE];

K M G
10 20 30

AU = 16b = 2B

PAGE = 128 KB = 64 KAU = 2^{16} AU OFF(16)

VAS = 8 GB = 4 GAU = 2^{32} AU VA(32)

PAS = 2 GB = 1 GAU = 2^{30} AU PA(30)

VA(32) = PAGE(16) : OFF(16)

PA(30) = FRAME(14) : OFF(16)

PD(16) = PRIV(2) : FRAME(14) ← 2B = 1AU.

```
int sharePages(PMT pmt1, ushort pg1, PMT pmt2, ushort pg2, ushort cnt){
    for (ushort i = 0; i < cnt; i++) {
        if (pmt1[pg1+i] != 0) return -1;
        if (pmt2[pg2+i] == 0) return -1;
        pmt1[pg1+i] = pmt2[pg2+i];
    }
    return 0;
}
```

2. (10 poena)

Neki sistem sa straničnom organizacijom virtuelne memorije i dohvatanjem stranica na zahtev (*demand paging*) ima PMT u dva nivoa, s tim da operativni sistem i PMT drugog nivoa takođe alocira na zahtev, tek kada proces prvi put ispravno pristupi nekoj stranici iz opsega te PMT drugog nivoa. Zato su inicijalno svi ulazi u PMT prvog nivoa *null*. Širina virtuelne adrese je 32 bita, stranica ima 4K adresibilnih jedinica, a PMT oba nivoa isti broj ulaza.

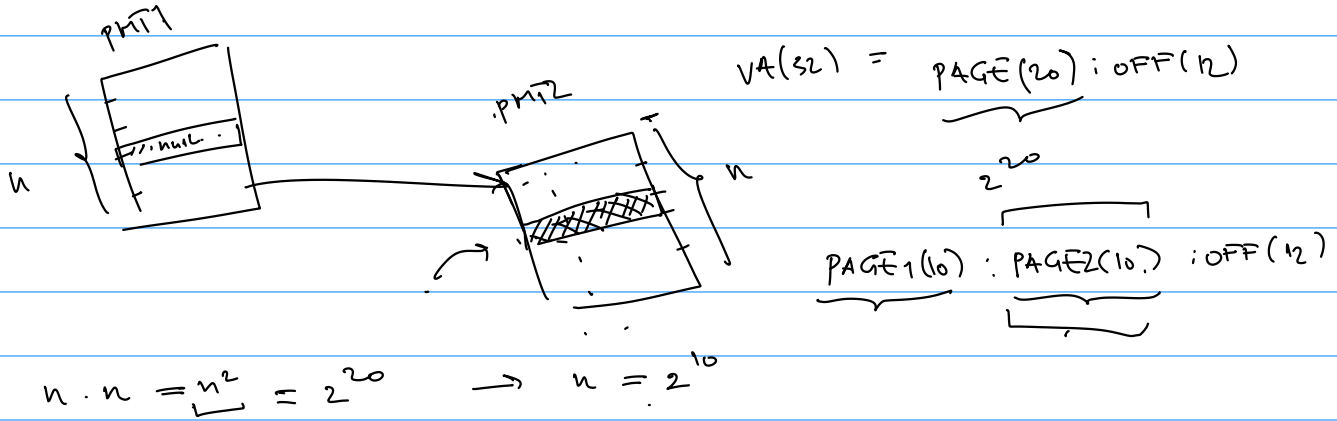
a)(7) Dole su date deklaracije tipova za stranice prvog i drugog nivoa. Implementirati operaciju `getPageDesc` koja za datu virtuelnu adresu vraća descriptor stranice u PMT kojoj ta virtuelna adresa pripada. U slučaju da PMT1 drugog nivoa nije alocirana, treba je alocirati. Za alokaciju memorije za svoje strukture kernel koristi svoju internu funkciju `kmalloc` koja je na raspolaganju. U slučaju da nema prostora za PMT1, funkcija `getPageDesc` treba da vrati *null*.

b)(3) Ukratko, ali precizno objasniti šta operativni sistem treba da uradi prilikom obrade stranične greške u opisanom sistemu.

```
const uint32 offsetw = 12;
const uint32 page1w = 10;
const uint32 PMT0_size = 1024;
const uint32 PMT1_size = 1024;

typedef uint32 PageDesc;
typedef PageDesc PMT1[PMT1_SIZE];
typedef PMT1* PMT0[PMT0_SIZE];
void* kmalloc (size_t size);

inline PageDesc* getPageDesc (PMT0 pmt, void* vaddr);
```




```
PageDesc * getPageDesc ( PMT0 pmt, void *vaddr ){
    uint32 va = (uint32)vaddr;
    uint32 page1 = va >> 22;
    uint32 pg2 = ( va << 10) >> 22;
    if ( pmt [page1] == 0 ) {
        pmt [page1] = kmalloc (1024 * sizeof(uint32));
        if ( pmt [page1] == null ) return null;
    }
    return & pmt [page1] [page2];
}
```

K M G T

AU = 1B

VAS = $16 \cdot 2^{60}$B = 2^{64}B

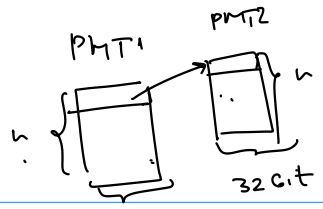
PAGE = 64KB = 2^{16}B

PAS = 4TB = 2^{42}B

$\frac{VAS}{PAGE} = \frac{2^{64}B}{2^{16}B} = 2^{48} = n \cdot n = n^2 \rightarrow [n = 2^{24}]$

VA(64) OFF(16)

[PA(42) = FRAME(26) : OFF(16)]



[VA(64) = PAGE1(24) : PAGE2(24) : OFF(16)]

page number

32 — 24 + 16 + 40

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 16EB (eksabajt, 1E=2^{60}) i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 64KB. Fizički adresni prostor je veličine 4TB (terabajt). PMT (*page map table*) je organizovana u dva nivoa, s tim da su i broj ulaza, kao i širina ulaza u PMT prvog i drugog nivoa isti (PMT oba nivoa su iste veličine). PMT oba nivoa smeštaju se u memoriju uvek poravnate na fizički okvir, odnosno uvek počinju na početku okvira. Zbog toga se u ulazu prvog nivoa čuva samo broj okvira u kom počinje PMT drugog nivoa, dok se preostali biti do celog broja bajtova u ulazu ne koriste; vrednost 0 u svim bitima označava hardveru da preslikavanje nije dozvoljeno ili moguće. U jednom ulazu PMT drugog nivoa čuva se broj okvira u koji se stranica preslikava i još 2 najniža bita koja koduju prava pristupa (00 – nedozvoljen pristup, stranica nije u memoriji, 01 – dozvoljeno samo izvršavanje instrukcije, 10 – dozvoljeno samo čitanje podataka, 11 – dozvoljeno i čitanje i upis podataka), dok se ostali biti ne koriste. Jedan ulaz u PMT prvog i drugog nivoa zauzima minimalan, ali ceo broj bajtova. Kada sistem kreira nov proces, ne učitava inicijalno ni jednu njegovu stranicu, niti alocira ijednu PMT drugog nivoa, već samo alocira PMT prvog nivoa, čije sve ulaze inicijalizuje nulama. Stranice se potom dohvataju na zahtev, tokom izvršavanja procesa, kada se po potrebi alociraju i PMT drugog nivoa.

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

Odgovor:

[fra- | **** | priv]

25 4 2

00 — ~~R~~

01 — X

10 — R

11 — RW

b)(7) Implementirati sledeću funkciju:

```
void releasePMTEntry (unsigned* pmt1, unsigned long page);
```

Ovu funkciju poziva kod kernela kada iz memorije izbacuje datu stranicu `page` procesa čija je PMT prvog nivoa zadata prvim argumentom. Ova funkcija treba da ažurira odgovarajuće ulaze u PMT (po potrebi, oba nivoa), s tim što treba i da dealocira PMT drugog nivoa, ako ona više nije potrebna. Na raspolaganju je interna funkcija kernela `dealloc_pmt(unsigned* pmt)` koja dealocira jednu PMT u memoriji. Ignorisati sve eventualne greške.

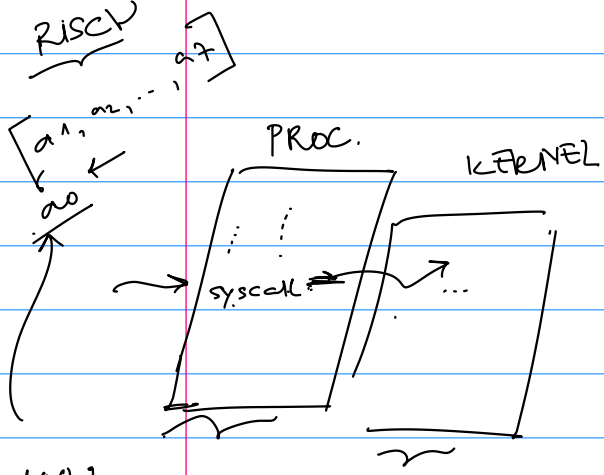
32 B

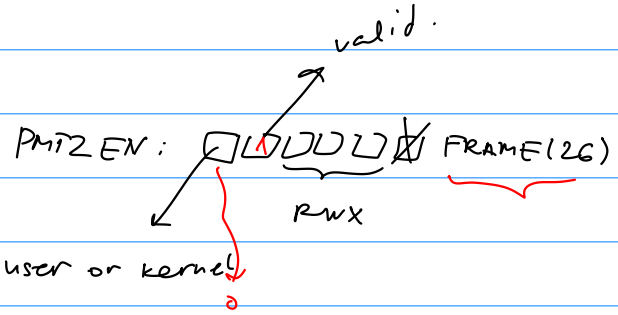
```
void releasePMTEntry (unsigned *pmt1, unsigned long page) {
    unsigned long page1 = page >> 40
    unsigned long page2 = (page << 24) >> 40;
    unsigned pmt1_entry = pmt1[page1];
    unsigned pmt2_frame = pmt1_entry >> 6.
    unsigned *pmt2 = (unsigned *)((unsigned long)pmt2_frame << 16)
    pmt2[page2] = 0;
```

1 AU 1L int 1UL

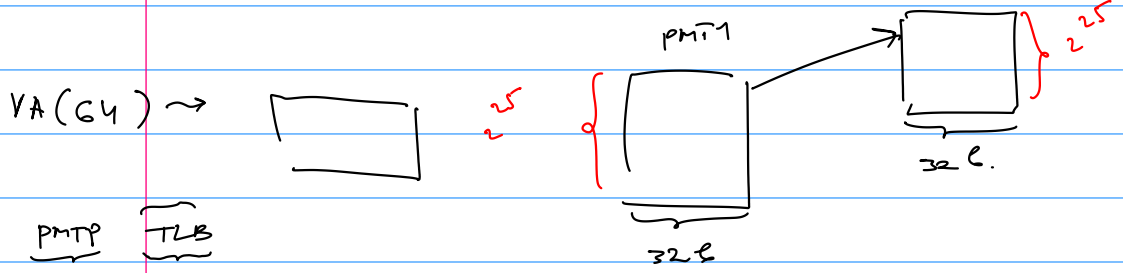
```
unsigned ptr2_size = 1U << 24;
int empty = 1;
for(unsigned i = 0; i < ptr2_size; i++){
    if (ptr2[i] ≠ 0){
        empty = 0
        break;
    }
}
if (empty) {
    dealloc_pt (ptr2);
}
}
```

RISC-V





1007



$PA(40) = FRAME(26) : OFF(14)$

32^e

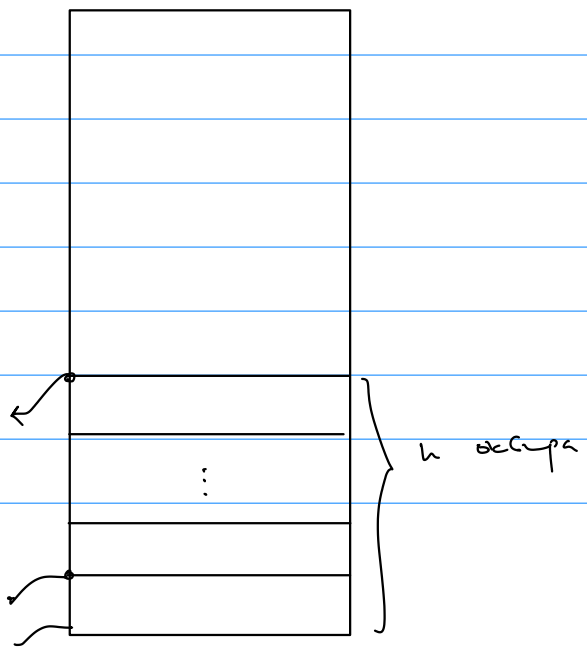
$VAS = 16 \cdot 2^{60} B = 2^{64} B$ $\quad VA(64)$

$PAGE = 16 KB = 2^{14} B$ $\quad OFF(14)$

$PAS = 1TB = 2^{40} B$ $\quad PA(40)$

$VA(64) = PAGE1(25) : PAGE2(25) : OFF(14)$

PAGE (50)



0

3. (10 poena)

Neki sistem zauzima najnižih n okvira raspoložive fizičke memorije za potrebe kernela. Kako ne bi menjao memorijski kontekst prilikom obrade sistemskih poziva, sistem preslikava (mapira) najviših n stranica svakog kreiranog procesa u ovaj memorijski prostor kernela, s tim da postavlja indikatore u deskriptoru tih stranica u PMT (*page map table*) tako da one nisu dostupne u neprivilegovanom (korisničkom) režimu rada procesora. Ostatak virtuelnog adresnog prostora dostupan je samom procesu za njegove potrebe i u korisničkom režimu.

Virtuelni adresni prostor je 16EB (eksabajt, 1E=2^{60}) i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 16KB. Fizički adresni prostor je veličine 1TB (terabajt). PMT je organizovana u dva nivoa, s tim da su i broj ulaza, kao i širina ulaza u PMT prvog i drugog nivoa isti (PMT oba nivoa su iste veličine). PMT oba nivoa smeštaju se u memoriju uvek poravnate na fizički okvir, odnosno uvek počinju na početku okvira. Zbog toga se u ulazu prvog nivoa čuva samo broj okvira u kom počinje PMT drugog nivoa, dok se preostali biti do celog broja bajtova u ulazu ne koriste; vrednost 0 u svim bitima označava da preslikavanje nije dozvoljeno. U jednom ulazu PMT drugog nivoa čuva se broj okvira u koji se stranica preslikava u najnižim bitima, dok 5 najviših bita ima sledeće značenje (tim redom, od najvišeg ka najnižem): bit koji govori da li je data stranica dostupna u neprivilegovanom (korisničkom) režimu rada procesora, bit koji govori da li je preslikavanje stranice moguće ili ne, i još 3 bita koja koduju prava pristupa (*rwx*), dok se ostali biti ne koriste. Jedan ulaz u PMT prvog i drugog nivoa zauzima minimalan, ali ceo broj bajtova.

32 − 5 − 26 = 1

Implementirati sledeću funkciju:

```
void initPMT (unsigned* pmt, unsigned long pageFrom, unsigned long nPages,
              unsigned long frameFrom, unsigned short rwx);
```

Ovu funkciju poziva kod kernela kada inicijalizuje PMT novokreiranog procesa, na čiju (već alociranu) PMT prvog nivoa ukazuje argument `pmt`, kako bi obavio opisano preslikavanje dela virtuelnog prostora u kernel prostor. Ona treba da preslika `nPages` susednih stranica počev od stranice `pageFrom` u isto toliko susednih okvira počev od okvira `frameFrom`. Argument `rwx` sadrži bite prava pristupa koje treba postaviti u deskriptore svih tih stranica.

PA

Na raspolaganju je interna funkcija kernela `alloc_pmt2` koja alocira jednu PMT drugog nivoa u memoriji, popunjava sve njene ulaze nulama i vraća broj okvira u kom počinje ta alocirana PMT. Pretpostaviti da će ova funkcija uvek uspeti da alocira PMT u memoriji (ignorisati greške). Veličine tipova su sledeće: `int` – 32 bita, `long` – 64 bita, `short` – 16 bita.

Obratiti pažnju na to da se ceo kernel kod izvršava u memorijskom kontekstu nekog procesa, što znači da su sve adrese (vrednosti pokazivača) virtuelne. Za konverziju neke fizičke adrese (vrednosti pokazivača) u kernel prostoru u virtuelnu adresu (u najvišem delu virtuelnog prostora svakog procesa), na raspolaganju je sledeća funkcija:

```
void* kaddrPtoV (void* physicalAddr);
```

```
void initPMT (unsigned* pmt, unsigned long pageFrom, unsigned long nPages,
              unsigned long frameFrom, unsigned short rwx) {

    for (unsigned long i = 0; i < nPages; i++) {
        unsigned long page = pageFrom + i;
        unsigned long page1 = page >> 39;
        unsigned long page2 = (page << 25) >> 39;
        if (pmt[page1] == 0) {
            unsigned long frame = alloc_pmt2();
            pmt[page1] = kaddrPtoV((void*)(frame << 14));
        }
        unsigned frame = frameFrom + i;
        unsigned pmt2_entry = ((unsigned)rwx << 27) | frame | (1u << 30);
        pmt[page1][page2] = pmt2_entry;
    }
```

}

}