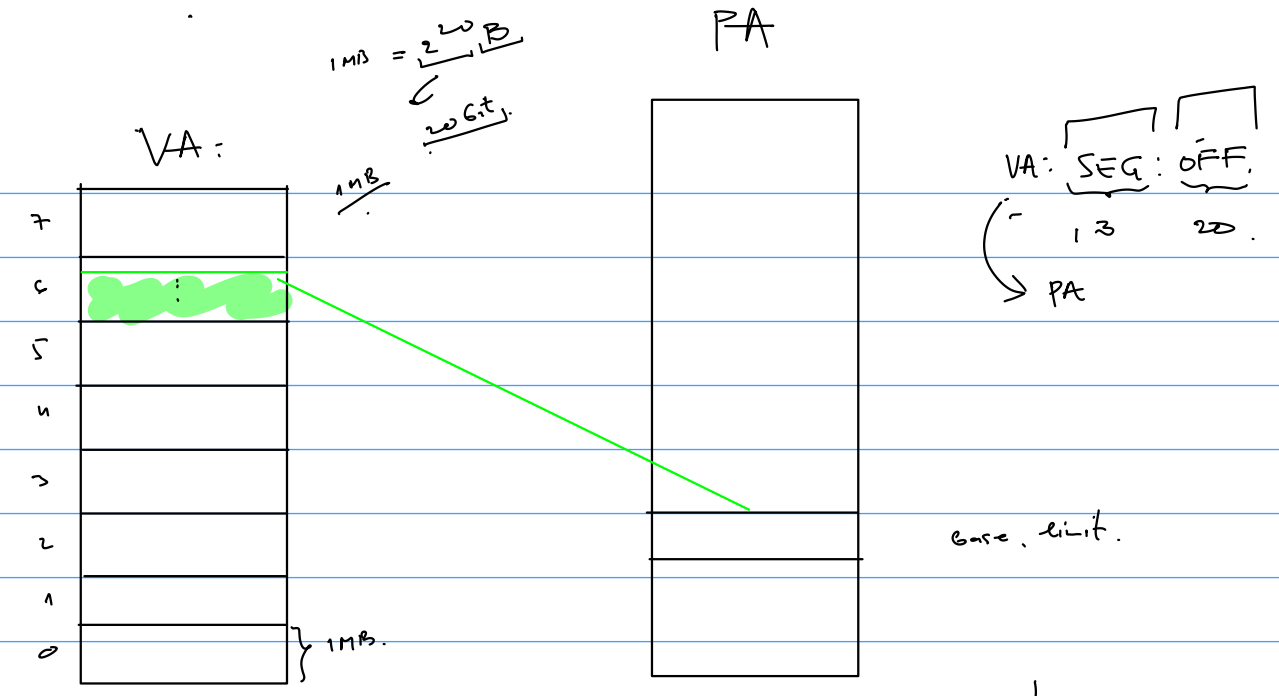


SMT — segment map table.

SMT?

VA → SEG. (0, 1, ..., 7) → SMTP + SEG · sizeof(SEG. DESC)

MMU

(11)

SEG. DESC: | base adress | limit.

SEG DESC 0
SEG DESC 1
ST.
ST
ST
STACK

{ limit > offset ✓
{ limit ≤ offset ↯ EXCEP.

base adress + offset → PA.

TEXT
TEXT
TEXT
code seg

offset ≤ limit → OK

offset > limit → NOT OK

1MB $2^{20} - 1$. 1...1 (20)

2^{20}B

2^{20} 21.

1 (20)

1TB = 2^{40} B

VA(32) = SEG(12) : OFF(20). 0...0

K 2^{10}

M 2^{20}

G 2^{30}.

1MB = 2^{20}B → 20

STACK	11 0
TEXT	00 1
DATA	11 0

VA(32) → 2^{32} B = 2^2 GB = 4 GB.

PA(40).

$\frac{4GB}{1MB} = \frac{4 \cdot 2^{30} B}{1 \cdot 2^{20} B} = 4 \cdot 2^{10} = 2^{12} \rightarrow 12$

1. (10 poena)

Neki sistem ima segmentnu organizaciju memorije. Virtuelna adresa je 32-bitna, adresibilna jedinica je bajt, a maksimalna veličina fizičkog segmenta je 1 MB. Fizički adresni prostor je veličine 1 TB. Jedan ulaz u SMT-u sadrži prava pristupa *rwx* u najniža tri bita, granicu fizičkog segmenta (*limit* u opsegu od 0 do maksimalne veličine segmenta minus 1) u bitima do njih, a zatim fizičku adresu u bitima do njih; ovaj deskriptor (ulaz u SMT-u) zauzima najmanji potrèban ceo broj bajtova.

Operativni sistem alocira segmente na zahtev, tako da pri kreiranju procesa ne alocira i ne učitava nijedan fizički segment. Vrednost 0 u polju za fizičku adresu u deskriptoru segmenta u SMT-u označavā da preslikavanje nije mogуće (segment nije alociran ili nije učitan).

Jedan alociran logički segment procesa opisan je deskriptorom tipa `SegDesc` u kom su, između ostalog, sledeća polja:

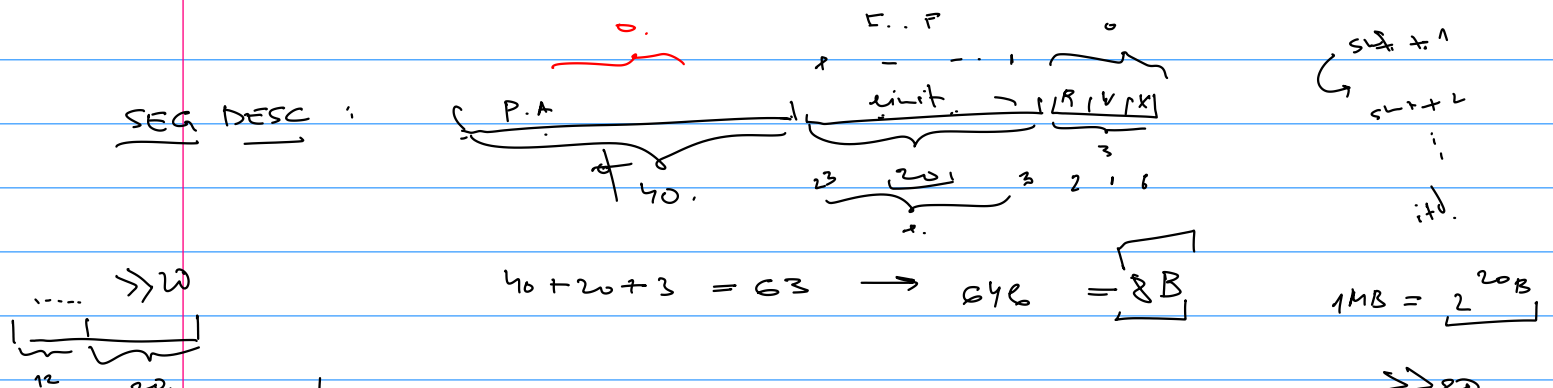
- `unsigned startAddr`: početna virtuelna adresa logičkog segmenta, svakako poravnata na početak fizičkog segmenta;
- `unsigned size`: veličina logičkog segmenta u bajtovima (može biti i veća od maksimalne veličine fizičkog segmenta);
- `unsigned short rwx`: prava pristupa za ceo logički segment u tri najniža bita.

Implementirati sledeću funkciju:

```
void initSegment (SegDesc* sd, unsigned long* smt);
```

Ovu funkciju poziva kod kernela kada inicijalizuje SMT novokreiranog procesa za svaki logički segment sa datim deskriptorom `sd`. Na već alociran SMT ukazuje `smt`. Veličine tipova su sledeće: `int` – 32 bita, `long` – 64 bita, `short` – 16 bita:

Rešenje:



40 + 20 + 3 = 63 → 64 = 8B

1MB = 2^{20}B

```
void initSegment (SegDesc *sd, ulong *smt){
    ulong smt_entry = 0;
    smt_entry |= sd->rwx;
    unsigned start_idx = startAddr >> 20;
    unsigned seg_num = size >> 20;          // broj celih segmenata
    unsigned seg_rem = (size << 12) >> 12;  // ostatak
    ulong off_mask = (ulong)(-1);
    off_mask = (off_mask >> 3) << 3;
    off_mask = (off_mask << 40) >> 40;
    ulong smt_entry_full = smt_entry | off_mask;
```

$2^8 = 16 \cdot 16 = 256$.

```
for (ulong i = 0; i < seg_num; i++) {
        sut[start_idx + i] = sut_entry_full;
}

if (seg_rem != 0) {
        sut[start_idx + seg_num] = (sut_entry | (seg_rem - 1) << 3);
}
}
```

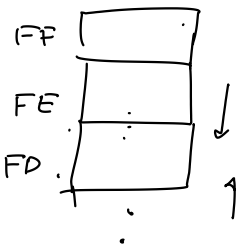
2. (10 poena)

U nekom sistemu sa segmentnom organizacijom memorije adresibilna jedinica je bajt, virtuelni adresni prostor je veličine 1 MB, a maksimalna veličina fizičkog segmenta je 4 KB. Dat je spisak početnih (virtuelnih) adresa, veličina (obe vrednosti su zapisane heksadecimalno) i vrsta logičkih segmenata (regiona) koje je alocirao neki proces.

Adresa segmenta (hex)	*Veličina (hex)*	*Vrsta segmenta*
0	2890	instrukcije
3000	FF0	konstantni podaci inicijalizovani statički
4000	3E68	instrukcije
28000	189A	promenljivi podaci
FE000	2000	stek

a)(7) U datu tabelu upisati parametre svih fizičkih segmenata koje je operativni sistem organizovao u SMT (sve ulaze u SMT koji nisu *null*) za ovaj proces, po rastućem redosledu broja segmenta. Broj segmenta i granicu segmenta (*limit*, najvišu dozvoljenu vrednost pomeraja u segmentu koji se sme adresirati) zapisati heksadecimalno, a bite prava pristupa *RWX* binarno tim redom. (Broj redova u datoj tabeli ne mora da odgovara broju segmenata koje treba upisati; eventualni višak redova date tabele ostaviti prazne.)

Segment # (hex)	*Limit (hex)*	*RWX (bin)*
0	FFF	001
1	FFF	001
2	88F	001
3	FEF.	100
4	FFF	001
5	FFF	001
6	FFF	001
7	E67.	001.
28	FFF	110
27	899	110
FE	FFF	110
FF	FFF	110.



b)(3) Ako stek raste ka nižim adresama, koja virtuelna adresa je prva koja je van dozvoljenog opsega i nije dozvoljena za adresiranje ako stek prekorači svoj dozvoljeni kapacitet?

Odgovor (*hex*): FDFFF.

Kom fizičkom segmentu pripada ta adresa?

Odgovor (*hex*): FD.

2. (10 poena)

U nekom sistemu adresibilna jedinica je bajt, virtuelni adresni prostor je veličine 4 GB, a fizički adresni prostor je veličine 1 GB. Sistem koristi segmentno-straničnu organizaciju memorije sa stranicom veličine 1 KB i segmentom maksimalne veličine 4 MB. Jedan ulaz u SMT zauzima dve 32-bitne reči; u nižoj je pokazivač (cela adresa) na PMT (ili 0 ako segment nije alociran), a u višoj reči su prava pristupa (RWX) u najniža tri bita i maksimalan dozvoljen broj stranice u segmentu (*limit*) u bitima do njih; preostali biti se ne koriste. Jedan ulaz u PMT zauzima jednu 32-bitnu reč (koriste se najniži biti).

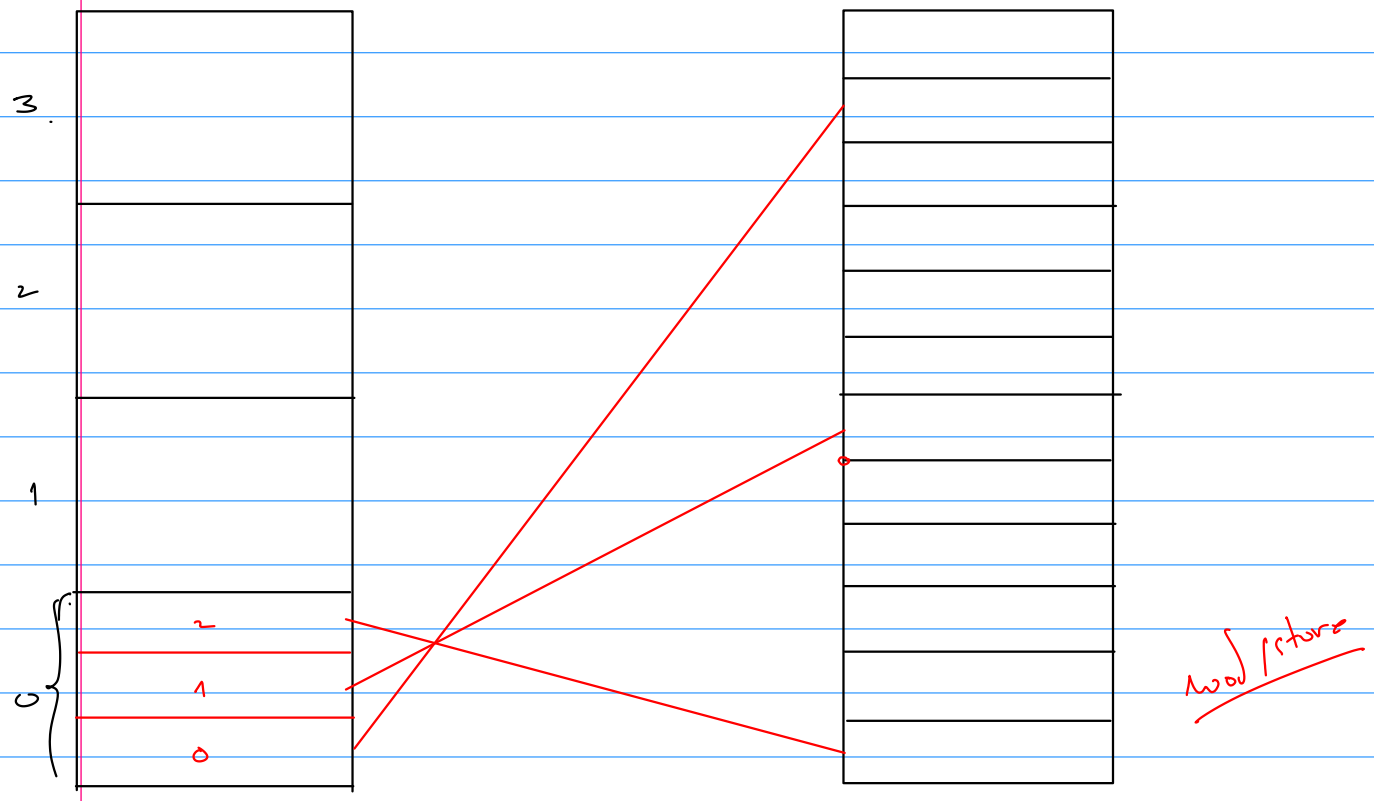
Da bi obradio sistemske pozive u kojima se neki parametar zadaje kao pokazivač (virtuelna adresa u adresnom prostoru pozivajućeg procesa), operativni sistem mora da konvertuje datu virtuelnu u fizičku adresu, jer se kod kernela izvršava u režimu bez preslikavanja adresa. Implementirati funkciju koja obavlja ovu konverziju (vraća *null* ako virtuelna adresa ne pripada alociranom delu virtuelnog adresnog prostora). Prvi parametar ove funkcije je pokazivač na SMT pozivajućeg procesa. Pretpostaviti da je `uint32` deklarisan kao neoznačeni 32-bitni celobrojni tip. Napisati sve potrebne deklaracije za tipove SMT i PMT.

```
void* v2pAddr (SMT pmt, void* vaddr);
```

Rešenje:

```
typedef uint32 PMT;
typedef uint64 SMT;

void *v2pAddr(SMT *smtp, void *vaddr) {
    PMT *pmtp = smtp[((uint32)vaddr) >> 22];
    if (pmtp == null) return null;
    PMT pd = pmtp[(((uint32)vaddr) << 10) >> 20];
    uint32 pa = pd + ((((uint32)vaddr) << 22) >> 22;
    return pa;
}
```



PMT - page map table.

page entry.

VA = SEG : PAGE : OFF.

O(1)

smtp + seg · sizeof(segdes)

pmtp + page size of(pmdes)

[SMTP]

seg desc: [PMTP] [priv];

MMU page desc: [frame]. → frame : offset

#include <stdio.h>

[printf(...);]

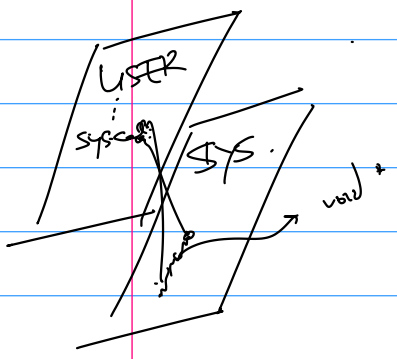
R1 R2 R3

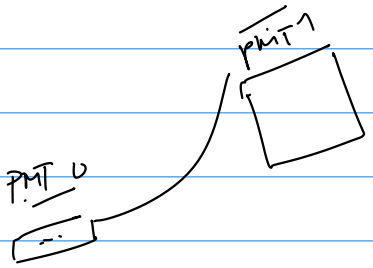
read(fd, size, &buffer);

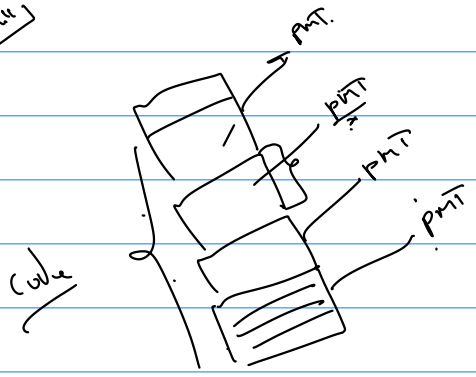
syscall 17

/ecall

void *







(~ 0)

null.

SMT ENTRY (64): [PMTP] [RWX] ... [PAGE NUM]

72 3 11..8..

PAGE ENTRY (32): [frame]

72

1 0 ... 0 -1

8

1 ... 1

1<<8

0, $2^8 - 1$

VA(32), PA(32)

SEG. SIZE: 16MB = 2^{24}B.

PAGE SIZE: 64KB = 2^{16}B.

PAGE PER SEG: $\frac{2^{24}B}{2^{16}B} = 2^8$

4GB = 2^{32}B

$2^8 \cdot 4KB = 1MB$

$\sim 10^8$

3. (10 poena)

Neki sistem ima segmentno-straničnu organizaciju memorije. Virtuelna adresa je 32-bitna, adresibilna jedinica je bajt, a maksimalna veličina fizičkog segmenta je 16 MB. Stranica je veličine 64 KB. Fizički adresni prostor je veličine 4 GB. Jedan ulaz u SMT-u zauzima dve 32-bitne reči; u prvoj (nižoj) su prava pristupa *rwx* u najviša tri bita, a granica fizičkog segmenta (*limit* u opsegu od 0 do maksimalno dozvoljenog broja stranice koji se sme adresirati) u najnižim bitima; druga (viša) reč ulaza u SMT-u sadrži 32-bitni pokazivač na PMT tog segmenta.

Operativni sistem alocira stranice na zahtev, tako da pri kreiranju procesa ne alocira nijednu stranicu. Vrednost 0 u polju za pokazivač na PMT u deskriptoru segmenta u SMT-u, kao i u deskriptoru stranice u PMT-u označava da preslikavanje nije moguće.

Jedan alociran logički segment procesa opisan je deskriptorom tipa `SegDesc` u kom su, između ostalog, sledeća polja:

- `unsigned startAddr`: početna virtuelna adresa logičkog segmenta, svakako poravnata na početak fizičkog segmenta;
- `unsigned size`: veličina logičkog segmenta u bajtovima (može biti i veća od maksimalne veličine fizičkog segmenta);
- `unsigned short rwx`: prava pristupa za ceo logički segment u tri najniža bita.

Implementirati sledeću funkciju:

```
int initSegment (SegDesc* sd, unsigned long smt[][2]);
```

Ovu funkciju poziva kod kernela kada inicijalizuje SMT novokreiranog procesa za svaki logički segment sa datim deskriptorom `sd`. Na već alociran SMT ukazuje `smt`. Potrebno je inicijalizovati ulaze u SMT i napraviti pridružene PMT za sve korišćene fizičke segmente. Veličine tipova su sledeće: `int` – 32 bita, `long` – 64 bita, `short` – 16 bita.

Statička funkcija `PMT::alloc()` alocira prostor u memoriji kernela za smeštanje PMT jednog segmenta, inicijalizuje ceo taj PMT na 0 i vraća 32-pokazivač taj PMT. Kako bi se obezbedilo da bude dovoljno prostora za sve PMT-ove potrebne za sve fizičke segmente koje zauzima jedan logički segment, funkcija `initSegment` treba najpre da proveri da li ima dovoljno prostora za njih. Ovu proveru obavlja statička funkcija `PMT::reserve(int segs)` koja proverava da li ima dovoljno prostora za PMT-ove `segs` segmenata, rezerviše taj prostor (da bi narednih `segs` poziva `PMT::alloc` sigurno uspelo) i vraća 0 u slučaju uspeha, a negativnu vrednost u slučaju nedostatka prostora. Funkcija `initSegment` treba da vrati 0 u slučaju uspeha, a negativnu vrednost u suprotnom.

```
int initSegmat (SegDesc *sd, unsigned int smt[][2]) {
    int segs = (sd->size) >> 24;
    int rem = ((sd->size) << 8) >> 8;
    if (rem != 0) segs++;
    int start_idx = (sd->startAddr) >> 24;
    if (PMT::reserve(segs) < 0) return -1;
    for (int i = start_idx; i < start_idx + segs; i++) {
        smt[i][0] = PMT::alloc();
        smt[i][1] = (int)(sd->rwx) << 29;
        smt[i][1] |= ((int)-1) << 8 - 1
    }
    if (rem != 0) {
        smt[start_idx + segs - 1] >>= 8;
        smt[start_idx + segs - 1] <<= 8;
        smt[start_idx + segs - 1] |= rem;
    }
```

```
    return 0;
}
```