Rešenja zadataka za
drugi kolokvijum iz Operativnih sistema 1
April 2019.

1. (10 poena)

```
class Mutex {
public:
  Mutex () : lck(0), holder(0), nestingCnt(0) {}
  int wait();
  int signal();
protected:
  void block (); // Implementacija ista kao i za Semaphore
  void deblock (); // Implementacija ista kao i za Semaphore
private:
  unsigned lck, nestingCnt;
  ThreadQueue blocked;
  Thread* holder;
};

int Mutex::wait () {
  if (holder==Thread::running) {
    nestingCnt++;
    return 0;
  }
  lock(&lck);
  if (holder) block();
  holder = Thread::running;
  nestingCnt++;
  unlock(&lck);
  return 0;
}

int Mutex::signal () {
  if (holder!=Thread::running || nestingCnt==0) return -1; // Error
  lock(&lck);
  if (--nestingCnt==0) {
    holder = 0;
    if (!blocked.isEmpty()) deblock();
  }
  unlock(&lck);
  return 0;
}
```

2. (10 poena)

a)(5)

```
void load_module (ModuleDesc* mod) {
  mem_extend(mod->size);  // Extend memory
  mod->base = (char*)0 + mem_size;
  mem_size += mod->size;
  load(mod->name,mod->base);
}
```

b)(5)

```
            load r1, [r0]     ; Load module's base address
            and  r1, r1, r1   ; If loaded,
            jnz  call_fun     ; call the subroutine
            push r0           ; Else, load the module
            call load_module
            pop  r0
            load r1, [r0]     ; Load module's base address
call_fun:   call #fun[r1]     ; Add the subroutine's offset
                              ; to the module's base addr and call it
```

3. (10 poena)

a)(2) Neka n označava broj bita za adresiranje ulaza u PMT svakog nivoa (broj ulaza u PMT svakog nivoa je 2^n), a neka je veličina stranice 2^p bajtova. Na osnovu uslova zadatka važi:
$n + n + p = 32$, jer je takva struktura 32-bitne virtuelne adrese;
$2^n \cdot 2^2 = 2^p$, odnosno $n + 2 = p$, jer je veličina jedne PMT jednaka veličini stranice.
Rešavanjem se dobija $n = 10$, $p = 12$, pa jedna PMT ima $2^{10} = 1024 = 1\text{K}$ ulaza, a stranica je veličine 4KB, koliko zauzima i jedna PMT prvog ili drugog nivoa.

b)(3) Prema tome, jedna PMT drugog nivoa „pokriva“ $1\text{K} \cdot 4\text{KB} = 4\text{MB}$ virtuelnog adresnog prostora. Zbog toga, za memorijski prostor kernela, procesi dele tačno jednu PMT drugog nivoa na koju ukazuje poslednji ulaz u PMT prvog nivoa i ta PMT drugog nivoa se ne alocira za svaki proces posebno.

PMT prvog nivoa ovog procesa, koja je svakako morala biti alocirana, zauzima 4KB. Za potrebe segmenata koda i podataka ovog procesa potrebna je i dovoljna samo jedna PMT drugog nivoa (1MB + 3MB = 4MB), a za segment steka još jedna. Tako je za PMT prvog i drugog nivoa ukupno alocirano tri stranice, odnosno 12KB memorije.

c)(2) Pošto su i PMT prvog i PMT drugog nivoa procesa roditelja i procesa deteta inicijalno potpuno iste, one se mogu u potpunosti deliti odmah nakon sistemskog poziva *fork* (ovi procesi imaju inicijalno isti i PMTP), i to sve dok neki od procesa ne izvrši upis u neku svoju stranicu. Prema tome, nije potrebno alocirati nikakav dodatni memorijski prostor za PMT procesa deteta.

d)(3) Kada jedan proces izvrši upis u jednu svoju stranicu, potrebno je „razdvojiti“, odnosno kopirati jednu PMT drugog nivoa procesa roditelja i procesa deteta. Osim toga, i odgovarajući ulazi u PMT prvog nivoa koji ukazuju na tu PMT drugog nivoa tako postaju različiti, jer moraju da ukazuju na različite PMT drugog nivoa, pa je potrebno „razdvojiti“, odnosno kopirati i PMT prvog nivoa. Zato je alocirano dve dodatne stranice, za smeštanje ove dve dodatne PMT, odnosno 8KB memorije.