Rešenja zadataka za
drugi kolokvijum iz Operativnih sistema 1
April 2017.

1. (10 poena)

```
struct ProcessorPrivate {
  int lock_count; // Counts the number of locks (nested critical sections)
  ...
  ProcessorPrivate () : lock_count(0),... {...}
};

void lock (short* lck) {
  disable_interrupts(); // Multiple calls of disable_intterupt are harmless
  processor_private[this_processor()].lock_count++;
  while (test_and_set(lck));
}

void unlock (short* lck) {
  *lck=0;
  if (--processor_private[this_processor()].lock_count==0)
    enable_interrupts();
}
```

2. (10 poena)

```
int extend (PCB* pcb, size_t by) {
  if (pcb==0) return -1; // Exception
  if (mem_allocate_at(by,pcb->base+pcb->size)) {
     // Extension without relocation
     pcb->size+=by;
     return 0;
  }
  void* newLocation;
  if (mem_allocate(pcb->size+by,&newLocation)) {
     // Extension with relocation
     mem_copy(pcb->base,newLocation,pcb->size);
     mem_free(pcb->base,pcb->size);
     pcb->base = newLocation;
     pcb->size+=by;
     return 0;
  }
  return -1; // No free memory, cannot extend
}
```

3. (10 poena)

a)(3) VA(64): Page1(25):Page2(25):Offset(14).
PA(40): Frame(26):Offset(14).

b)(7)

```
const unsigned short pg1w = 25, pg2w = 25, offsw = 14;

inline unsigned getPMT1Entry (unsigned long vaddr) {
  return vaddr>>(pg2w+offsw);
}

inline unsigned getPMT2Entry (unsigned long vaddr) {
  return (vaddr>>offsw) & ~(-1L<<pg2w);
}

int getPMTEntry (unsigned* pmt1, unsigned long vaddr, unsigned* value) {
  unsigned pmt1entry = getPMT1Entry(vaddr);
  if (pmt1[pmt1entry]==0){
    *value = 0;
    return -1;
  }
  unsigned* pmt2 = (unsigned*)(((unsigned long)pmt1[pmt1entry])<<offsw);
  unsigned pmt2entry = getPMT2Entry(vaddr);
  *value = pmt2[pmt2entry];
  return (*value&3==0)?-1:0;
}

void setPMTEntry (unsigned* pmt1, unsigned long vaddr, unsigned value) {
  unsigned pmt1entry = getPMT1Entry(vaddr);
  if (pmt1[pmt1entry]==0)
    pmt1[pmt1entry] = alloc_pmt();
  unsigned* pmt2 = (unsigned*)(((unsigned long)pmt1[pmt1entry])<<offsw);
  unsigned pmt2entry = getPMT2Entry(vaddr);
  pmt2[pmt2entry] = value;
}

int sharePage (unsigned* pmtFrom, unsigned long pageFrom,
      unsigned* pmtTo, unsigned long pageTo){
  unsigned long vaddrFrom = pageFrom<<offsw;
  unsigned long vaddrTo = pageTo<<offsw;
  unsigned value;
  if (getPMTEntry(pmtFrom,vaddrFrom,&value)<0)
    return -1;
  setPMTEntry(pmtTo,vaddrTo,&value);
  return 0;
}
```