Prvi kolokvijum iz Operativnih sistema 1
Septembar 2013.

1. (10 poena)

```
static int completed = 0;
static int status = 0;
static REG* ptr = 0;
static unsigned int count = 0;

void transfer (IORequest* ioHead) {
  while (ioHead) {
    completed = 0;  // initialize transfer
    status = 0;
    ptr = ioHead->buffer;
    count = ioHead->size;
    *ioCtrl = 1; // Start I/O
    while (!completed); // wait for I/O to complete
    ioHead->status=status;  // set status
    ioHead->callBack(ioHead);  // signal completion
    ioHead = ioHead->next;  // take next
  }
}

interrupt void ioInterrupt () {
  if (*ioStatus&2) { // Error in I/O
    completed = 1;
    status = -1;
    *ioCtrl = 0; // Stop I/O 2
    return;
  }
  *ptr++ = *ioData; // input data
  if (--count == 0) { // transfer completed
    completed=1;  // signal completion
    *ioCtrl = 0;    // stop I/O
  }
}
```

2. (10 poena)

a)(5) VA: Segment(8):Page(16):Offset(8); PA: Frame(20):Offset(8).

b)(5) FF00FFh

3. (10 poena) a)(7)

```
gcd:        LOAD  R0,#x[SP]   ; R0:=x
            LOAD  R1,#y[SP]   ; R1:=y
            CMP   R1,#0       ; if (y==0)
            JMPNE gcd_else
gcd_then:   POP   PC          ; return x (already in R0)
gcd_else:   PUSH  R1          ; push y for call of remainder
            PUSH  R0          ; push x for call of remainder
            CALL  remainder
            POP   R1          ; remove x from the stack
            POP   R1          ; remove y from the stack
            PUSH  R0          ; push remainder(...) for call of gcd
            PUSH  R1          ; push y for call of gcd
            CALL  gcd
            POP   R1          ; stack cleanup
            POP   R1
            POP   PC          ; return (the result is already in R0)
```

b)(3) Jeste bezbedna (ovakvi potprogrami se ponekad nazivaju *reentrant*), pošto uopšte ne pristupa statičkim (globalnim) podacima, već sve podatke ima ili na steku, ili u registrima, što je deo konteksta niti.

4. (10 poena)

```
void visit_node (void* n) {
  if (n==0) return;
  Node* nd = (Node*)n;
  process(&nd->data);
  if (nd->children_head) {
    for (Node* p=nd->children_head->next_sibling; p; p=p->next_sibling)
      thread_create(&visit_node,p);
    visit_node(nd->children_head);
  }
}
```