Prvi kolokvijum iz Operativnih sistema 1
Jun 2019.

1. (10 poena)

```
static REG* ioPtr = 0;
static int ioCount = 0;
static int ioCompleted = 0;

void transfer (int count) {
  REG* buffer = new REG[count];

  // I/O 1
  ioPtr = buffer;
  ioCount = count;
  *io1Ctrl = 1; // Start I/O 1
  while (ioCount>0) {
    while (!(*io1Status&1)); // busy wait
    *ioPtr++ = *io1Data;
    ioCount--;
  }
  *io1Ctrl = 0; // Stop I/O 1

  // I/O 2:
  ioPtr = buffer;
  ioCount = count;
  ioCompleted = 0;
  *io2Ctrl = 1; // Start I/O 2

  // Wait for I/O 2 completion:
  while (!ioCompleted);

  delete [] buffer;
}

interrupt void io2Interrupt() {
  *io1Data = *ioPtr++;
  if (--ioCount == 0) {
    ioCompleted = 1;
    *io2Ctrl = 0; // Stop I/O 2
  }
}
```

2. (10 poena)

```
int dispatch (Thread* newT = 0) {
  lock();
  if (newT && !newT->isRunnable) {
    unlock();
    return -1;
  }
  int oldR = Thread::running;
  jmp_buf oldC = Thread::allThreads[oldR].context;
  if (newT)
    Thread::running = newT - Thread::allThreads;
  else
    do
      Thread::running = (Thread::running+1)%NumOfThreads;
    while (!Thread::allThreads[Thread::running].isRunnable &&
            Thread::running!=oldR);
  if (Thread::running!=oldR) {
    jmp_buf newC = Thread::allThreads[Thread::running].context;
    yield(oldC,newC);
  }
  unlock();
  return 0;
}
```

3. (10 poena)

```
struct copy_task {
  void* dst;
  const void* src;
  size_t num;
};

void copy_block (void* task) {
  copy_task* t = (copy_task*)task;
  memcpy(t->dst,t->src,t->num);
}

void* par_memcpy (void* dest, const void* source, size_t num) {
  if (num<=0) return dest;
  int tail = num%BLOCK_SIZE;
  int numOfBlocks = num/BLOCK_SIZE + (tail?1:0);
  copy_task* tasks = new copy_task[numOfBlocks];
  for (int i=0; i<numOfBlocks; i++) {
    tasks[i].dst = (char*)dest + i*BLOCK_SIZE;
    tasks[i].src = (const char*)source + i*BLOCK_SIZE;
    tasks[i].num = (i<numOfBlocks-1 || tail==0) ? BLOCK_SIZE : tail;
    thread_create(&copy_block,&tasks[i]);
  }
  wait(NULL);
  return dest;
}
```