Prvi kolokvijum iz Operativnih sistema 1
Odsek za softversko inženjerstvo
Mart 2016.

1. (10 poena)

```
IORequest *dma1Pending = 0, *dma2Pending = 0; // Currently pending requests

void startDMA1 () {
  if (ioHead!=0 && dma1Pending==0) {
    dma1Pending = ioHead; // Take the first request,
    ioHead = ioHead->next; // remove it from the list
    if (ioHead==0) ioTail = 0;
    *dma1Ctrl = dma1Pending->buffer; // and assign it to DMA1
    *dma1Ctrl = dma1Pending->size;
    *dma1Ctrl = 1; // Start I/O
  }
}

void startDMA2 () {
  if (ioHead!=0 && dma2Pending==0) {
    dma2Pending = ioHead; // Take the first request,
    ioHead = ioHead->next; // remove it from the list
    if (ioHead==0) ioTail = 0;
    *dma2Ctrl = dma2Pending->buffer; // and assign it to DMA2
    *dma2Ctrl = dma2Pending->size;
    *dma2Ctrl = 1; // Start I/O
  }
}

void transfer (IORequest* req) {
  req->next = 0;
  if (!ioHead) {
    ioHead = ioTail = req;
    startDMA1();
    startDMA2();
  } else
    ioTail = ioTail->next = req;
}

interrupt void dmaInterrupt () {
  if (*dma1Status&1) { // DMA1 transfer complete
    if (dma1Pending==0) return; // Exception
    if (*dma1Status&2) // Error in I/O
      dma1Pending->status = -1;
    else
      dma1Pending->status = 0;
    dma1Pending = 0;
    startDMA1();
  }
  if (*dma2Status&1) { // DMA2 transfer complete
    if (dma2Pending==0) return; // Exception
    if (*dma2Status&2) // Error in I/O
      dma2Pending->status = -1;
    else
      dma2Pending->status = 0;
    dma2Pending = 0;
    startDMA2();
  }
}
```

2. (10 poena)

```
dispatch:   ; Save the current context
            load  rx,running
            store r0,#offsR0[rx] ; save regs
            store r1,#offsR1[rx]
            ...
            store r31,#offsR31[rx]
            store pc,#offsPC[rx] ; save pc
            store psw,#offsPSW[rx] ; same psw
            store sp,#offsSP[rx] ; save sp

            ; Select the next running process
            call  scheduler

            ; Restore the new context
            load  rx,running
            load  sp,#offsSP[rx] ; restore sp
            load  psw,#offsPSW[rx] ; restore psw
            load  pc,#offsPC[rx] ; restore pc
            load  r0,#offsR0[rx] ; restore regs
            load  r1,#offsR1[rx]
            ...
            load  r31,#offsR31[rx]
            ; Return
            iret
```

3. (10 poena)

```
// Wrapper, for type-casting only:
inline void* _createSubtree (void* n) {
  return createSubtree(*(int*)n);
}

Node* createSubtree (int n) {
  if (n<=0) return 0;
  Node* node = new Node();
  if (n<=1) return node;

  int n1 = n-1;
  // Create a new thread to create the right subtree:
  pthread_t pid;
  pthread_create(&pid,&_createSubtree,&n1);
  // Create the left subtree in this thread:
  node->leftChild = createSubtree(n1);
  // Join with the right-subtree child thread:
  pthread_join(pid,&node->rightChild);
  return node;
}
```