Prvi kolokvijum iz Operativnih sistema 1
Odsek za računarsku tehniku i informatiku
Maj 2015.

1. (10 poena)

```
static IORequest* pending[2] = {0,0}; // Pending requests for two channels

void startIO (int i) { // Helper: start a new transfer with channel i
  if (ioHead==0) {
    *ioCtrl &= ~(1<<i); // Stop channel i
    return;
  }
  pending[i] = iohead;
  ioHead = ioHead->next; // Remove the request from the list
  // Start I/O with channel i:
  if (pending[i]->dir)
    *ioCtrl |= 1<<(2+i);
  else
    *ioCtrl &= ~(1<<(2+i));
  *ioCtrl |= (1<<i);
}

void handleInterrupt (int i) { // Helper: handle interrupt from channel i
  if ((*ioStatus)&(1<<(2+i)))
    pending[i]->status = -1;  // Error in I/O
  else {  // Transfer the next data item
    if (pending[i]->dir)
      ioData[i] = *(pending[i]->buffer)++;
    else
      *(pending[i]->buffer)++ = ioData[i];
    if (--pending[i]->size)
      return;
    else
      pending[i]->status = 0;  // Transfer completed successfully
  }
  startIO(i); // Initiate the next transfer with this channel
}

void transfer () {
  startIO(0);
  startIO(1);
}

interrupt void ioInterrupt () {
  if (*ioStatus & 1)
    handleInterrupt(0);
  else
    handleInterrupt(1);
}
```

2. (10 poena)

a)(7)

```
sys_call:   ; Save the current context
            push  r0 ; save r0 temporarily on the stack
            load  r0,userRunning
            store r1,#offsR1[r0] ; save regs
            pop   r1
            store r1,#offsR0[r0]
            store r2,#offsR2[r0]
            ...
            store r31,#offsR31[r0]
            pop   r1 ; save pc
            store r1,#offsPC[r0]
            pop   r1 ; save psw
            store r1,#offsPSW[r0]
            pop   r1 ; save original sp
            store r1,#offsSP[r0]

            ; Restore the new context
            load  r0,kernelRunning
            load  r1,#offsSP[r0] ; restore original sp
            push  r1
            load  r1,#offsPSW[r0] ; restore original psw
            push  r1
            load  r1,#offsPC[r0] ; restore pc
            push  r1
            load  r1,#offsR1[r0] ; restore regs
            load  r2,#offsR2[r0]
            ...
            load  r31,#offsR31[r0]
            load  r0,#offsR0[r0]
            ; Return (but stay in kernel mode)
            iret
```

b)(3) Procedura `switch_to_user` izgleda potpuno analogno (skoro potpuno isto) kao i data procedura `sys_call`, samo što promenljive `userRunning` i `kernelRunning` zamenjuju mesta (uloge), a neposredno pre instrukcije `iret` stoji još samo 1instrukcija `setusr`.

3. (10 poena)

a)(5) Problem je u neispravnim kontrolnim strukturama u telu funkcije `pipe`. Početna (roditeljska) nit izvršava *then* granu prve *if* naredbe, u kojoj se poziva funkcija `writer` i koja se onda neograničeno (beskonačno) izvršava, pa ta prva nit nikada ne izlazi iz ove funkcije. Nit-dete, kreirana u prvom *fork* pozivu, izvršava *else* granu iste prve *if* naredbe, u kojoj se poziva funkcija `reader` koja se takođe neograničeno izvršava. Prema tome, ni jedna od ove dve niti neće nikada doći do druge *if* naredbe, pa se drugi par niti nikada neće kreirati.

b)(5)

```
void pipe () {
  for (int i=0; i<N; i++)
    if (fork())
      writer(&c[i],&flag[i]);
    else
      if (fork())
        reader(&c[i],&flag[i]);
}
```