Prvi kolokvijum iz Operativnih sistema 1
Odsek za softversko inženjerstvo
Mart 2019.

1. (10 poena)

```
void performIO () {
  while (ioHead!=0) {

    IORequest* ioPending = ioHead; // Take the first request,
    ioHead = ioHead->next; // remove it from the list,
    if (ioHead==0) ioTail = 0;

    *ioBlock = ioPending->block; // and read it from I/O
    *ioCtrl = START_READING;

    while (*ioStatus&1==0);  // Wait for completion

    if (*ioStatus&2) // Error in I/O
      ioPending->status = -1;
    else {
      ioPending->status = 0;
      for (int i=0; i<BLOCK_SIZE; i++)
        ioPending->buffer[i] = *ioData;
    }
  }
}

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

2. **(10 poena)**

```
dispatch:   ; Save the current context
            push  r0    ; save r0 temporarily on the stack
            load  r0, [running] ; r0 now points to the running PCB
            store r1, #offsR1[r0] ; save r1
            store r2, #offsR2[r0] ; save r2
            ...                   ; save r3-r30
            store r31, #offsR31[r0] ; save r31
            pop   r1                ; save r0
            store r1, #offsR0[r0]
            pop   r1                ; save PC
            store r1, #offsPC[r0]
            pop   r1                ; save PSW
            store r1, #offsPSW[r0]
            pop   r1                ; save SP
            store r1, #offsSP[r0]

            ; Select the next running process
            call  scheduler

            ; Restore the new context
            load  r0, [running]
            load r1, #offsSP[r0] ; restore SP
            push r1
            load r1, #offsPSW[r0] ; restore PSW
            push r1
            load r1, #offsPC[r0] ; restore PC
            push r1
            load r31, #offsR31[r0] ; restore R31
            ...                    ; restore r30-r2
            load r1, #offsR1[r0] ; restore R1
            load r0, #offsR0[r0] ; restore R0

            ; Return
            iret
```

3. (10 poena)

```
const int N = ...;
const unsigned timeout = 5000;
int pid[N];

int main (int argc, const* char argv[]) {
  if (argc<2) {
    printf(“Error: Missing argument for the program to run.\n”);
    exit(-1);
  }

  int ret = 0;  // Status to return on exit

  // Create children:
  for (int i=0; i<N; i++) {
    pid[i] = fork();
    if (pid[i] < 0) {
      printf(“Error: Failed to create a child process number %d.\n”,i);
      ret = -2;
    } else
    if (pid[i] == 0) {
      execlp(argv[1]);
      printf(“Error: Failed to execute the program for the child process
number %d.\n”,i);
      exit(-1);  // Terminate the child, not the parent
    }
  }

  // Wait for all children:
  int stat = wait(NULL,timeout);
  if (stat<0) {
    printf(“Error: Failed to wait for the children processes.\n”);
    ret -= 4;
  }

  if (stat == 0) {
    printf(“All children completed in time.\n”,i);
    exit(0);
  }

  // Kill all incomplete children:
  for (int i=0; i<N; i++) {
    stat = wait(pid[i],0);
    if (stat == 0) continue;  // Child completed
    if (stat<0) {
      printf(“Error: Failed to wait for the child process number %d.\n”,i);
      if (ret > -8) ret -= 8;
    }
    stat = kill(pid[i]);
    if (stat<0) {
      printf(“Error: Failed to kill the child process number %d.\n”,i);
      if (ret > -16) ret -= 16;
    }
  }

  exit(ret);
}
```