Test pitanja za vežbu u vezi bloka tema 9 i 10

1. Označiti tačne iskaze u vezi promena taska kod kooperativnih i preemptive RTOS:
-kod kooperativnog RTOS se selektovanje aktivnog taska vrši iz tekućeg taska
-kod kooperativnog RTOS prelaz između taskova je moguć i nakon timeout-a
-kod preemptive RTOS selektovanje aktivnog taska se vrši forsirano, od strane operativnog sistema
-kod preemptive RTOS prelazi između taskova su unapred poznati
-kod preemptive RTOS promena taska može biti inicirana nekim hardverskim događajem
-kod kooperativnog RTOS promena taska može biti inicirana i drugim načinima osim funkcije OS za -promenu taska
-kod preemptive RTOS promena taska može biti inicirana nekim vremenskim intervalom
-kod kooperativnog RTOS promena taska može biti aktivirana mutexom

2. Označi tačne tvrdnje:
-Kod kooperativnog RTOS se zbog načina promene aktivnog taska može javiti problem sa stekom
-Kod preemptive RTOS je zbog načina promene aktivnog taska unapred poznata alokacija stek memorije
-Kod kooperativnog RTOS je u određenim slučajevima moguće imati jedinstvenu stek memoriju
-Kod kooperativnog RTOS je uvek obavezno imati posebne stek memorije za svaki task
-Kod preemptive RTOS je uvek moguće imati jedinstvenu stek memoriju
-Kod preemptive RTOS je obavezno imati posebne stek memorije za svaki task

3. Označiti netačnu tvrdnju/tvrdnje za kooperativne RTOS u odnosu na softverske FSM:
-lakše pisanje programa (ali paziti na međuzavisnosti taskova)
-veći izlazni kod
-eventualni problemi sa stekom
-sporije pisanje programa

4. Pronađi uljeza/uljeze među elementima sinhronizacije i komunikacije među taskovima:
-semafor
-bafer
-*mailbox*
-red čekanja (*queue*)
-*mutex*
-*deadlock*

5. Upariti vrste RTOS sa objašnjenjem
Hard RTOS - potencijalno katastrofalni scenariji u slučaju otkaza
Firm RTOS - moguć neadekvatan odziv sa malom verovatnoćom, ali nije katastrofalni scenario
Soft RTOS - moguće smanjenje perfomansi sistema, ali nije kritično i dozvoljeno je.

6. Označi netačan razlog/razloge za uvođenje RTOS u odnosu na programiranje bez OS (*baremetal*):
-rešavanje zadataka koji imaju vremensko ograničenje na realizaciju odziva
-bolja kontrola neadekvatnog rada ili otkaza
-štednja memorije
-lakša realizacija pseudo-paralelnog rada

7. Označi netačnu tvrdnju/tvrdnje o tome šta preemptive RTOS uvodi kao dodatne mogućnosti u odnosu na kooperativne:
-Aktiviranje taska iz prekidne funkcije sistemskog tajmera
-Aktiviranje taska nakon nekog događaja
-Prioritet taskova
-Timeout taskova
-Round-robin metoda za izbor narednog taska

8. Označiti stanje/stanja taska za koja je odmah moguće preći u taj task i aktivirati ga
-Aktivan (Active)
-Spreman (Ready)
-U čekanju (Wait)
-Suspendovan (Suspended)

9. Na slici su data 4 taska koji su ili kooperativni ili preemptive. Označi onaj task/taskove koji su kooperativni:

```
void Task1(void)
{
  static char led;
  while (1)
  {
    RED = ~led;
    OS_Sleep(250); //sačekaj i promeni task
    led++;
  }
}
```

```
void Task2()
{
    while(1)
    {
        printf("Ovo je Task 2 \n");

        //wait
        Sleep(1000);//1 sec

        //start the task
        StartTask(TaskId_1);
    }
}
```

```
void task3(void)
{
  while(1)
  {
    OS_Switch();
    ....
  }
}
```

```
// Toggles the LED at pin 9 every time the push button is pressed.
void task4(void *p)
{
  unsigned char flag=HIGH;
  while(1)
  {
    OSTakeSema(&sem);
    digitalWrite(9, flag);
    flag=!flag;
  }
}
```

10. Označiti moguća stanja taska:
Aktivan (Active) – task je aktivan (upravo se izvršava)
Spreman (Ready) – task nije aktivan, ali je spreman za rad i može se aktivirati.
U čekanju (Wait) – task je u stanju čekanja na neki događaj ili istek vremenskog intervala i nije spreman za aktiviranje.
Suspendovan (Suspended)

11. Za sledeći primer main funkcije jednostavnog kooperativnog RTOS označiti redove u kojima se vrši promena taska:

```
1: void main(void)
2:{
3:  OSInit(0);
4:  OSCreateTask(Task1);
5:  OSCreateTask(Task2);
6:  OSCreateTask(Task3);
7:  OSRun();
8:}
```

-4
-5
-6
-7
-nije prikazan mehanizam promene taska

12. Označiti tačne tvrdnje za Preemptive operativne sisteme:

a) U ovakvim sistemima je predvidiv trenutak poziva funkcije za promenu taska
b) Nužno je obezbediti poseban stek za svaki task
c) Ovakvi OS se retko primenjuju na mikrokontrolere sa malo RAM memorije
d) U Preemptive OS se koristi jedinstven stek

13. Označi tačne iskaze vezane za Cooperative i Preemptive RTOS :

a) Cooperative RTOS - selektovanje aktivnog taska se vrsi pozivom odgovarajuce funkcije operativnog sistema
b) Preemptive RTOS - selektovanje aktivnog taska se vrsi iz tekuceg taska
c) Preemptive RTOS - selektovanje aktivnog taska je inicirano nekim hardverskim ili softverskim dogadjajem
d) Cooperative RTOS - selektovanje aktivnog taska se vrsi od strane operativnog sistema

14. Označiti jedan ili više tačnih iskaza koji se odnose na Preemptive operativni sistem:
a) svaki task ima TimeOut
b) kad se task proglasi aktivnim, njegov TCB se prebacuje na početak liste
c) ukoliko postoji više taskova sa najvišim prioritetom, aktivni task se najčešće određuje Round Robin metodom
d) kod principa ulančane liste, analiza pri izboru narednog aktivnog taska počinje od prvog narednog taska

15. Dopuni :
Odziv koji je duži od zadatih vremenskih ograničenja na odziv dovodi kod:
a) Hard RTOS do ..... sistema. Tolerancija na ovakav odziv je minimalna ili nikakava.
b) Firm RTOS do ..... neadekvatnog odziva ukoliko je verovatnoća pojave dugačkog odziva mala.
c) Soft RTOS do ..... perfomansi sistema, ali to nije kritično i dozvoljeno je.
d) Navigacija helikoptera, navigacija raketa, robotizovana hirurgija je primer za .....

e) Navigacija robota koji uklanja korove u poljoprivredi je primer za .....
f) Navigacija robota koji uklanja korove u poljoprivredi je primer za .....

16. Zaokruži tačne tvrdnje koje se odnose na prioritet izvršavanja taskova:
1. Prioritet, kao dodatni parametar, može se koristiti samo ako je podržan od strane operativnog sistema
2. Istovremeno može biti aktivirano više taskova, ako su oni istog prioriteta
3. Ako je više taskova u wait stanju, u naredno stanje active preći će onaj sa najvišim prioritetom
4. Ako ima više taskova spremnih za rad, sa istim prioritetom, najčešće se primenjuje Round Robin metoda

17. Zaokružiti mane RTOS-a u odnosu na FSM:
1. teže pisanje programa
2. veći izlazni kod
3. eventualni problemi sa stekom
4. teži posao za programera

18. Prelaz iz jednog taska u drugi (zaokružiti odgovore koji su tačni):
a) Moguće je ostvariti primenom Round Robin metode, pri tome će se taskovi u Ready stanju i istog prioriteta naizmenično aktivirati
b) Moguć je ako je task u stanju Ready
c) Ostvaruje se pozivanjem određene sistemske funkcije
d) Moguć je ako je task najvišeg prioriteta, i pri tome u stanju Wait

19. Označi rešenja za problem primene multitasking operativnog sistema na mikrokontrolerima sa malo RAM memorije koji potiče od metode preklapanja (overlay) lokalnih promjenljivih:
1. Sve lokalne promenljive treba deklarisati kao static.
2. Umesto lokalnih promenljivih koriste se globalne promenljive.
3. Umesto lokalnih promenljivih koriste se globalne promenljive u formi unija, koje omogućavaju programirano preklapanje za funkcije koje nisu istovremeno aktivne.
4. Ako to kompajler dozvoljava, svaki task treba definisati kao posebno stablo poziva, pri čemu se funkcija taska nalazi na dnu ovog stabla.

20. Koje su prednosti RTOS :

-Povećava pouzdanost sistema
-Za sinhronizaciju zadataka se ne koriste semafori
-Smanjuje se cena razvoja (skraćuje se vreme realizacije)
-Povećava se cena razvoja (povećava se vreme realizacije)
-RTOS ne garantuje da ce krajni rokovi vremenski kritičnih zad. biti ispoštovani u svim uslovima
- Olakšana prenosivost (prenos aplikacija na drugu platformu)

21.

```
void Task1(void)
{
  while (1)
  {
      ...
      OS_Switch();
```

```
      ...
      func_t1();
      ...
   }
}
void func_t1(void)
{
 ...
 OS_Switch();
 ...
}

void Task2(void)
{
  while (1)
  {
      ...
      OS_Switch();
      ...
      func_t2();
      ...
   }
}
void func_t2(void)
{
  ...
 OS_Switch();
 ...
}
```

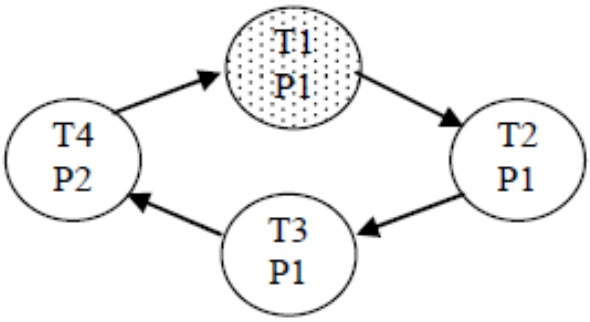
Stek
S3 ret_2
S2 ret_t1
S1 SP

Za dati primer programa sa dva taska zaokružiti tačne tvrdnje:

a) Predvidjeni tok izvršavanja programa se ne narušava i sa steka se povlači adresa povratka u Task1 (ret_t1)
b) Predvidjeni tok izvrsavanja programa se narusava i sa steka se povlaci adresa povratka u Task2 (ret_t2)
c) Problem steka za promenu aktivnog taska se moze prevazici ako se funkcije za promenu aktivnog taska pozivaju isključivo u glavnoj funkciji
d) Funkcije za promenu aktivnog taska mogu se pozivati i iz funkcija nižeg nivoa (funkcija koje su pozvane iz funkcije taska) bez ikakvih problema

22. U sledećem primeru taskovi T1, T2, T3 imaju niži prioritet (P1) u odnosu na task T4 koji ima viši prioritet (P2). Neka su u datom primeru taskovi T1, T2, T3 uvek u stanju Ready, odnosno koriste Time

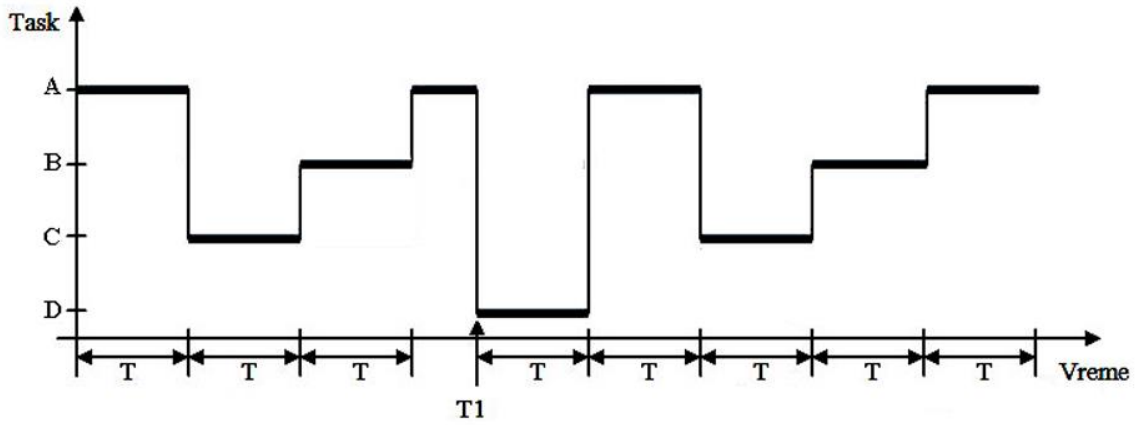
Slice za svoj rad (periodično aktiviranje), a neka task T4 reaguje na spoljašnje hardverske impulse. (Za analizu ovog zadatka ne koristiti princip ulančane liste)



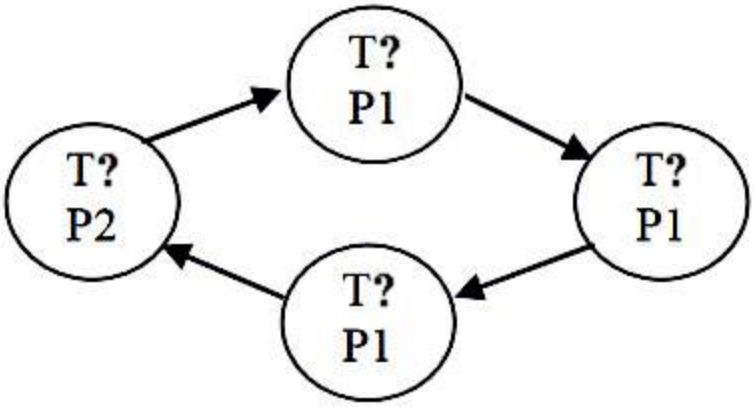
Zaokružiti moguć/moguće raspored/rasporede aktiviranja taskova:

A) T1 - T4 - T1 - T4 - T1
B) T1 - T2 - T4 - T3 - T1
C) T1 - T2 - T3 - T1 - T2
D) T1 - T1 - T2 - T3 - T4

23. Na osnovu datog vremenskog prikaza raspodele taskova zameniti "?" indeksom odgovarajućeg task-a.



Smenjivanje taskova se vrši Round Robin metodom. A, B, C imaju isti prioritet i uvek su u stanju ready, dok je task D višeg prioriteta i reaguje na određen hardverski impuls koji se desio u trenutku T1.



24. Odabrati moguće rešenje/rešenja problema ilustrovanog sledećim primerom ako je drugi parametar funkcije OS_WaitSem alternativno vreme čekanja nakon čijeg isteka task može biti aktiviran i ako se odgovarajući semafor nije aktivirao (0 znači da čekanje na semafor nije ograničeno):

```
void Task1(void)
{
    ...
    while(1)
    {
        ...
        OS_WaitSem(ev_1, 0);
        OSSetSem(ev_2);
        ...
    }
}

void Task2(void)
{
    ...
    while(1)
    {
        ...
        OS_WaitSem(ev_2, 0);
        OSSetSem(ev_1);
        ...
    }
}
```

a) definisanje semafora u formi unije
b) promena vrednosti drugog parametra funkcije OS_WaitSem
c) zamena redosleda funkcija OS_WaitSem i OSSetSem u Task1
d) problem se ne može rešiti ni na jedan od ponuđenih načina