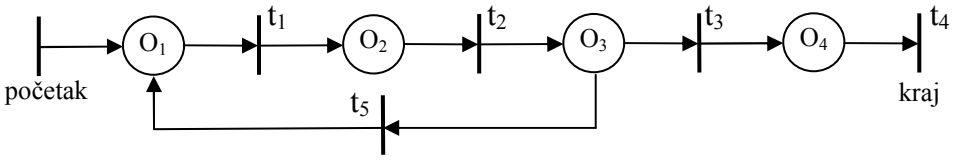
Završni ispit iz predmeta *SUSTAVI ZA RAD U STVARNOM VREMENU i neki odgovori* 1.7.2010.

(svaki zadatak nosi 3 boda)

1. Neki proces sastoji se od slijedećih operacija: $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4$. Svaka od operacija traje od 50 do 100 ms. Ponekad (iznimno) operacija O_3 zahtijeva ponavljanje operacija $O_1 \rightarrow O_2$ (tj. jedan niz može izgledati: $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4$). Petri mreža za zadani proces nacrtana je ispod. Napisati tablicu s vremenima tranzicija.



2. Zadan je sustav zadataka: τ_1: T_1=10, c_1=5; τ_2: T_2=15, c_2=5; τ_3: T_3=20, c_3=5; τ_4: T_4=25, c_4=5. Navesti implicitne trenutke krajnjeg završetka (*implicitni deadline*) za sve zadatke. Grafički prikazati raspoređivanje koje koristi RMPA metodu za zadani sustav zadataka na dvoprocesorskom računalu. Stati s raspoređivanjem kada se potvrdi rasporedivost sustava ili nemogućnost njegova rasporeda.

3. Zadan je sustav zadataka:

   τ_1: T_1=10, c_1=5; τ_2: T_2=15, c_2=2; τ_3: T_3=30, c_3=2; τ_4: T_4=40, c_4=2.

   Korištenjem alternativnog kriterija rasporedivosti potvrditi da je sustav rasporediv po RMPA metodi. Za pronađene l i k obrazložiti značenje korištene nejednakosti.

$$\left(\min\left\{ \frac{1}{l \cdot T_k} \sum_{j=1}^{i} c_j \left\lceil \frac{l \cdot T_k}{T_j} \right\rceil \;\middle|\; (k,l) \in R_i \right\} \right) \le 1$$

$$R_i = \left\{ (k,l) \;\middle|\; k \in 1 \ldots i \;\wedge\; l \in 1 \ldots \left\lfloor \frac{T_i}{T_k} \right\rfloor \right\}$$

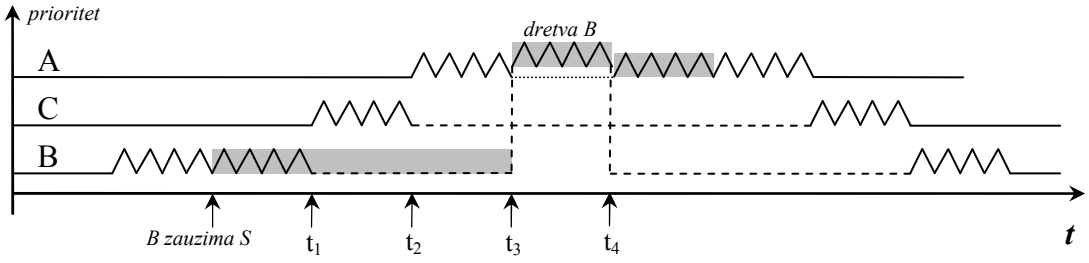
   Obrazloženje l i k: traži se onaj segment vremena koji je višekratnik periode zadatka k, a u kojem se (počevši u kritičnom slučaju) mogu obaviti svi zadaci koji se u tom periodu javljaju i to onoliko puta koliko se ti zadaci javljaju u tom periodu.

4. U sustavu koji koristi *DDS* (raspoređivanje prema krajnjim trenucima završetka) pojavljuju se događaji za koja su poznata i vremena izračunavanja, kao i krajnji trenuci završetka:

   (*vrijeme pojave*, *potrebno procesorsko vrijeme*, *krajnji trenutak dovršetka*) = (t_i, c_i, k_i) =
   { (1, 3, 5), (2, 1, 4), (4, 2, 7), (6, 2, 9), (8, 3, 11), (13, 1, 20) }.

   Provjeriti je li slijed događaja rasporediv *DDS* postupkom na dvoprocesorskom računalu.

5. Dretve A i B, različita prioriteta, koriste isto sredstvo S prema slici ispod. Osim njih u sustavu se pojavljuje i dretva C (u t_1). Opisati što se dogodilo u trenucima t_1, t_2, t_3 i t_4. Sustav koristi raspoređivanje prema prioritetu.



   u t3 se pojavljuje inverzija prioriteta te se primjenjuje *protokol nasljeđivanja prioriteta*

6. Sinkronizacijski mehanizmi su često neophodni u višedretvenom programu. Međutim, njihova uporaba mora biti pažljivo analizirana jer može uzrokovati mnoge probleme. Koji su to problemi? Kako ih se može riješiti/izbjeći/ublažiti?

   potpuni zastoj, inverzija prioriteta

7. U RT sustavu koji koristi kružno posluživanje (SCHED_RR) pojavljuju se slijedeći nezavisni poslovi: (*id_posla, vrijeme_pojave, trajanje, prioritet*) = { (1, 0, 6, 5), (2, 2, 6, 5), (3, 5, 6, 5), (4, 9, 5, 10), (5, 12, 2, 1) }. Veći broj predstavlja veći prioritet. Kvant vremena iznosi jednu jedinicu vremena (kućanski se poslovi zanemaruju). Prikazati redoslijed odvijanja poslova na procesoru dok se svi poslovi ne obave do kraja.

```
01234567890123456789012345678901 (vrijeme)
11212312344444123123233555------
11121213244444132323233555------ ili (ako novi ide na kraj reda)
```

8. Straničenje („virtualna memorija“) kao način upravljanja spremnikom ima mnoge prednosti naspram ostalih metoda upravljanja spremnikom. Ipak, rijetko se koristi u sustavima za rad u stvarnom vremenu. Zašto?

ako se koristi disk kako pomoćni spremnik, promašaj u programu uzrokuje predugi zastoj u izvođenju programa (mora se stranica dohvatiti s diska)

u jednostavnijim sustavima, sklop za straničenje može znatno povećati cijenu procesora

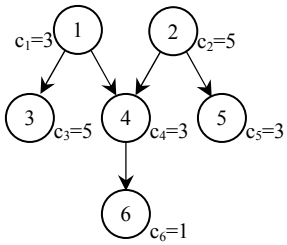
9. Pri sumiranju ogromne matrice (NxN), program koristi slijedeći C odsječak (N mod 4 = 0!):

```
suma = 0;
for ( i = 0; i < N; i = i + 4 )
     for ( j = 0; j < N; j++ )
          suma += A[i][j] + A[i+1][j] + A[i+2][j] + A[i+3][j];
```

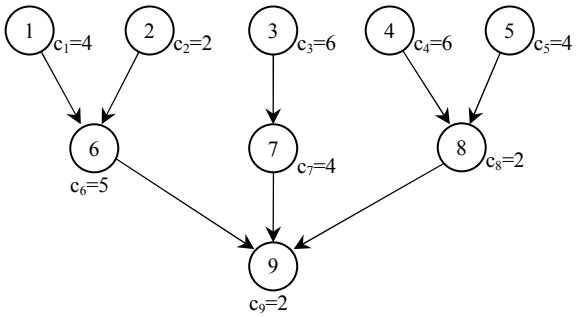
Iako je broj iteracija petlji manji (broj zbrajanja je ipak jednak), kod je lošiji od slijednog (umjesto `i = i + 4` → `i++`, a umjesto duge sume samo `suma += A[i][j]`. Zašto?

sa i+4 se previše „šara“ po memoriji pa se priručna memorija ne koristi efikasno, a u sustavima sa straničenjem može se znanto povećati broj promašaja

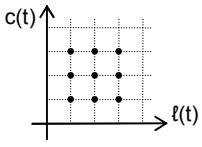
10. Korištenjem općeg raspoređivanja (GS) napraviti raspored sustava zadataka koji je zadan necikličkim računalnim grafom (desno). Raspoređivanje napraviti za dva procesora. Prikazati mogući raspored zadataka po procesorima korištenjem rezultata općeg raspoređivanja (napraviti PS - *preemptive scheduling*).



11. Sustav zadataka zadan je usmjerenim grafom (slika ispod). Korištenjem postupka raspoređivanja sa stablenom strukturom (*rooted computation tree*) izgraditi računalno stablo. Na stablu naznačiti visine (h). Izračunati trajanje izračunavanja cijelog sustava zadataka raspoređenog ovom metodom za slučajeve kada u sustavu postoje dva procesora.



12. Sustav zadataka u trenutku t=0 prikazan je ℓ/c grafom (desno). Koliko je minimalno procesora potrebno da sustav bude rasporediv ukoliko se koristi EDF (te LLF kao 2. kriterij kada prema EDF-u ima više kandidata od raspoloživih procesora)? Pokažite da je manji broj procesora (za jedan) nedostatan. Kolika je procesorska zalihost u prvih pet jedinica vremena.



treba 4 procesora, zalihost je 3 (4*5-(3*1+3*2+2*3)-(5-3))

13. Pri sinkronizaciji s poslužiteljem klijent je primio odgovor s vremenskim oznakama (t_1, t_2, t_3, t_4) = (50, 140, 155, 155) (t_1 – vrijeme slanja zahtjeva (klijentsko vrijeme), t_2 – vrijeme primanja zahtjeva (poslužiteljsko vrijeme), t_3 – vrijeme slanja odgovora (poslužiteljsko vrijeme), t_4 – vrijeme primanja odgovora (klijentsko vrijeme)). Ukoliko je poznato da je veza asimetrična te slanje poruke (od strane klijenta) traje duplo duže od primanja iste, odrediti broj milisekundi koji treba dodati klijent na svoj sat da bi se uskladio sa satom poslužitelja.

ukupno trajanje „putovanja paketa“ = $(t_4–t_1) – (t_3–t_2)$ = (155–50) – (155–140) = 105–15 = 90
obzirom da je veza asimetrična i prema poslužitelju putuje duplo duže, očito su trajanja 60 i 30
trenutak primitka poruke kod poslužitelja, u vremenu klijenta je t_1+60 = 50+60 = 110, a trebalo bi biti 140
prema tome, klijent treba dodati 30 ms na svoj sat da bi se uskladio s poslužiteljem

14. Korištenje Interneta kao infrastrukture ili samo „njegovih“ protokola nije preporučljivo za RT sustave zbog mnogih njegovih nedostataka u tom okruženju. Ipak, kada nema alternative, koji se njegovi mehanizmi i protokoli mogu iskoristit u te svrhe? Obrazložiti.

    Koje prednosti ima raspodijeljena RT aplikacija koja koristi infrastrukturu Interneta u odnosu na sličnu koja koristi zasebnu (izdvojenu) vezu između čvorova RT sustava (a zašto je Internet, tj. njegova preteča i napravljena)?

protokoli: IP, UDP, RTP/RTCP – protokoli koji pružaju više kontrole na upravljanjem veze; TCP ne!
prednosti:
- robusnost, ako neka veza zakaže postoje druge (redundancija u spojnim putevima)
- jeftinije

15. Usporedbom dva algoritma utvrđeno je slijedeće: prosječno trajanje prvog je 100ms uz standardnu devijaciju od 1ms te prosječno trajanje drugog od 70ms uz standardnu devijaciju od 10ms. Koji je od njih bolji izbor za RT sustav? Obrazložiti!

Uobičajeno razmišljanje „gotovo sve spada u npr. 3*std.dev. pa je *drugi* s 'najdužim' vremenom od 100 bolji od *prvog* koji ima 103“ nije ovdje dobro, već treba obuhvatiti SVE slučajeve, tj. treba uzeti i one slučajeve koji se javljaju znatno rjeđe – uzeti npr. 5*std.dev.: najgore za prvi je 105, a za drugi 120. Još rjeđi slučajevi su značajno gori po drugi algoritam, ali se u nekom RT sustavu koji dulje radi ipak javljaju i mogli bi uzrokovati probleme ako se nije uzelo u obzir njihovo trajanje.