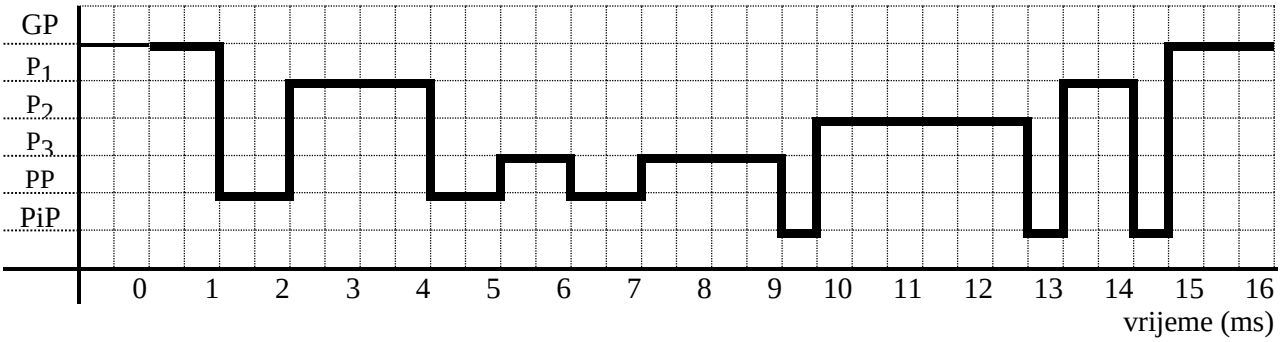
5. 7. 2024.	Ime i prezime	JMBAG
Operacijski sustavi, pismeni ispit		

1. (6) U nekom sustavu bez sklopa za prihvat prekida ali s programskom potporom za prihvat prekida prema prioritetu javljaju se prekidi: P1 u trenutku t=1 ms, P3 u t=4 ms, te prekid P2 u t=6 ms. Prioritet prekida određen je brojem (P3 ima najveći prioritet). Grafički prikazati aktivnosti procesora u glavnom programu (GP), procedurama za obradu prekida (Pi) koje traju 3 ms te procedurama za prihvat prekida (PP) koje traju 1 ms i povratak iz prekida (PiP) koje traju 0,5 ms.



Navesti vrijednosti svih struktura podataka koje se koriste u trenutku t=10 ms.

TP=2 KZ=000 KON[1]={0, kon(GP)} KON[2]={1, kon(P1)} KON[3]={-,-}

2. (6) Neka naprava koristi sklop za izravni pristup spremniku (DMA). Prosječno u sekundi prenosi 40000 podataka (svaki preko zasebnog sabirničkog ciklusa). Nakon što se prenese 1000 podataka sklop izaziva prekid. Obrada takvog prekida potroši 250 sabirničkih ciklusa. Koji postotak sabirničkih ciklusa treba navedena naprava ako sabirnica radi na 1 MHz?

prekida u sekundi: 40000/1000 = 40 => traju 40*250 = 10000 => uz 40000 za prijenos to je 50000

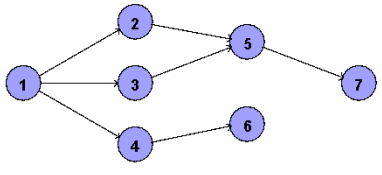
50000 / 1000000 * 100 % = 5 %

3. (6) U promatranom trenutku stanje sustava je sljedeće: dretva 7 je aktivna; dretve 6, 5 i 1 su u redu pripravnih dretvi, dretva 4 je u redu općeg semafora S te dretva 3 je u redu odgođenih – treba još 15 otkucaja sata da se od tamo makne. Svi redovi organizirani su po prioritetu gdje veći broj dretve označava veći prioritet (osim odgođenih i redova naprava). Pozivi koji se tada događaju su redom, jedan nakon drugoga zadani u donjoj tablici. Pokazati stanje sustava nakon svakog od ovih poziva.

	Početno stanje	Nakon `ZapočniUI(1)`	Nakon `PostaviOSEM(S)`	Nakon `Odgodi(17)`	Nakon `Prekid_sata()`
`Aktivna_D`	7	**6**	**6**	**5**	**5**
`Pripravne_D`	6 5 1	**5 1**	**5 4 1**	**4 1**	**4 1**
`Odgođene_D`	3^{15}	$\mathbf{3^{15}}$	$\mathbf{3^{15}}$	$\mathbf{3^{15}6^2}$	$\mathbf{3^{14}6^2}$
`OSEM[S]`	4	4	**- (.v=0)**	**- (.v=0)**	**- (.v=0)**
`Red UI[1]`	-	**7**	**7**	**7**	**7**

4. (6) Zadan je sustav zadataka Z_1-Z_7. Ako se zadaci izvode slijedno ($Z_1 \rightarrow Z_2 \rightarrow ... \rightarrow Z_7$) rezultate je ispravan. Korištenjem tablice odrediti maksimalno paralelni sustav zadataka. Sinkronizirati sustav općim semaforima. Ako svaki zadatak traži I*100 ms procesorskog vremena (I je indeks zadatka), koliko će na četvero-procesorskom sustavu trajati izvođenje sustava zadataka?

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
M1	D			K		K	
M2	K	D	D	D			D
M3		K			D		
M4			K		K		K

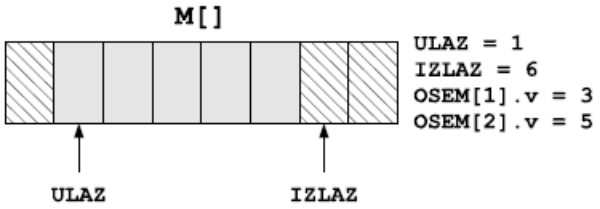


Z1: T1; PostaviOSEM(a); PostaviOSEM(b); PostaviOSEM(c);
Z2: ČekajOSEM(a); T2; PostaviOSEM(d);
Z3: ČekajOSEM(b); T3; PostaviOSEM(d);
Z4: ČekajOSEM(c); T4; PostaviOSEM(e);
Z5: ČekajOSEM(d); ČekajOSEM(d); T5; PostaviOSEM(f);
Z6: ČekajOSEM(e); T6;
Z7: ČekajOSEM(f); T7;

trajanje: najdulji put 1-2-5-7 => (1+3+5+7)*100 = 1600 ms = 1,6 s

5. (6) Napisati kod i navesti potrebne strukture podataka za komunikaciju više proizvođača i više potrošača preko ograničenog međuspremnika, uz sinkronizaciju semaforima.

(iz skripte),



Slika 6.4. Ograničeni međuspremnik

- `M[N]` – međuspremnik s `N` mjesta za poruke
- `ULAZ` i `IZLAZ` se povećavaju po modulu `N`
- `OSEM[1]` – broji poruke u međuspremniku, početno 0
- `OSEM[2]` – broji prazna mjesta u međuspremniku, početno `N` (veličina međuspremnika)

6.1.6. Više proizvođača i više potrošača

Potrebno je dodati dva binarna semafora: jedan za proizvođače da se spriječi istovremeni pokušaj stavljanja u međuspremnik i promjena varijable `ULAZ`, te jedan za potrošače da se spriječi pokušaj istovremenog uzimanja poruke iz međuspremnika i mijenjanja varijable `IZLAZ`.

```
proizvođač
{
    ponavljaj {
        P = stvori poruku()
        Čekaj_OSEM(2)
        Čekaj_BSEM(1)
        M[ULAZ] = P
        ULAZ = (ULAZ + 1) MOD N
        Postavi_BSEM(1)
        Postavi_OSEM(1)
    }
    do zauvijek
}
```

```
potrošač
{
    ponavljaj {
        Čekaj_OSEM(1)
        Čekaj_BSEM(2)
        P = M[IZLAZ]
        IZLAZ = (IZLAZ + 1) MOD N
        Postavi_BSEM(2)
        Postavi_OSEM(2)
        obradi poruku(P)
    }
    do zauvijek
}
```

6. (6) U nekom sustavu se izvode dretve 1a, 1b, 2a, 2b, 3a, 3b. Sve se raspoređuju prema prioritetu kao primarnom kriteriju te redu prispijeća kao sekundarnom (način SCHED_FIFO). Prioritet dretvi određen je njenim indeksom - 3a i 3b imaju najveći prioritet 3. Dretve se javljaju u sustavu u trenutcima: (t, D) = {(0, 2a), (1, 1a), (2, 2b), (3, 3a), (5, 3b), (9, 1b)}. Pokazati izvođenje dretvi do t=20 ukoliko svaka dretva treba četiri jedinice vremena za svoje izvođenje.

Aktivna	2a	2a	2a	3a	3a	3a	3a	3b	3b	3b	3b	2a	2b	2b	2b	2b	1a	1a	1a	1a	1b
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

7. (6) U sustavu koji koristi statičko upravljanje spremnikom s 8 MB spremnika i dvije particije, jedna od 3 MB i druga od 5 MB, javljaju se zahtjevi za pokretanjem procesa te završetka prema redu zadanom u donjoj tablici. Plus ispred oznake procesa +PX označava zahtjev za pokretanje procesa PX koji treba X MB radnog spremnika, dok –PX označava kraj izvođenja tog procesa. Procesi P1-P3 su pripremljeni za manju particiju, a ostali za veću. Prikazati promjene stanje spremnika za zadane zahtjeve. Zahtjevi koji se ne mogu poslužiti čekaju trenutak kad se budu mogli i tad se poslužuju.

zahtjev	radni spremnik - 8 MB = 3 + 5	
+P4		P4
+P2	P2	P4
–P4	P2	
+P1	P2	
+P5	P2	P5
–P2	P1	P5
+P3	P1	P5
–P1	P3	P5
+P4	P3	P5
–P3		P5
+P2	P2	P5

P1 čeka (P1 mora na početak particije, ne može uz P2 što su mnogi napravili)

P1 kreće
P3 čeka
P3 kreće
P4 čeka

8. (4) Za neki proces zadana je tablica prevođenja (desno). Ako je veličina stranice 64 KB (2^{16} B) kako će se prevesti logičke adrese (ako će se prevesti):

str.	okvir	bit pris.
0	0x0016	1
1	-	0
2	0x0023	1
3	0x0048	1
4	-	0

`0x`**`0000`**`1234 =>` `0x`**`0016`**`1234`

`0x`**`0002`**`3458 =>` `0x`**`0023`**`3458`

`0x`**`0004`**`5678 =>` prekid – stranica nije u memoriji

`0x`**`0056`**`7890 =>` prekid – stranica ne postoji (ni u tablici prevođenja - OS prekida proces)

16 bita za odmak = 4 heksa znamenke

9. (6) U nekom algoritmu potrebno je zbrojiti matrice A i B (A+=B). Ukoliko su matrice velike (N*N), a veličina stranice je dovoljna za pohranu jednog retka, odrediti koliko će promašaja generirati taj algoritam ako za elemente matrica su na raspolaganju dva okvira. Algoritam zamjene stranica je LRU.

```
za i=1 do N
   za j=1 do N
         A[i,j] += B[i,j]
```

stranice 1-N => matrica A
stranice N+1, N+N => matrica B
Za svaki „i“ trebaju nam samo redci „i“ od matrica A i B, tj. stranice „i“ i „i+N“.
Tj. za zbrajanje jednog retka imamo dva promašaja.
Ukupno je to 2N promašaja.

10. (6) Neka datoteka velika 2 MB kompaktno je pohranjena na disku od bloka 100001. Na tom disku se koristi UNIX i-node datotečni sustav, veličina bloka od 1 KB i veličine kazaljke 64 bita. Pokazati strukturu podataka koje opisuju smještaj te datoteke (sadržaj koji je poznat upisati u nju).

2 MB / 1 KB = 2*1024 KB/1 KB = 2048 blokova nam treba na disku: 100001 je prvi, 102048 je zadnji
u jedan blok stane: 1 KB/64 bita = 1024 * 8 bita / 64 bita = 128 kazaljki
2048 blokova datoteke opisuje isto toliko kazaljki koje su raspoređene:
10 u samom opisniku; ostaje 2048 – 10 = 2038
11. kazaljka pokazuje na blok s 128 idućih kazaljki; ostaje 2038 – 128 = 1910 => 14 * 128 + 118
12. kazaljka pokazuje na blok s kazaljkama, u tom bloku od 128 kazaljki koristi se samo prvih 15 koje pokazuju na blokove s kazaljkama: prvih 14 se koriste do kraja (svih 128), a u 15. prvih 118
skica: opisnik: [100001, 100002, ..., 100010, X, Y, -] (prvih 10 + 11. + 12.)
blok X na disku: [100011, 100011, ..., 100138]
blok Y na disku: [B1, B2, ..., B14, B15]
u blokovima B1-B15 (sumarno) kazaljke: [100139, ..., 102048]

11. (4) Za neki poslužitelj potrebno je 20 TB korisnog kapaciteta diskova. Ukoliko na raspolaganju stoji 10 diskova, svaki kapaciteta 5 TB, koji RAID odabrati (0,1,5,6) i koliko diskova spojiti (1-10), a da bi sustav bio „najsigurniji“ (i da ima barem 20 TB korisnog prostora)? Obrazložiti.

odabrati RAID 6 zbog otpornosti na kvara dva diska

odabrati minimalan broj diskova koji zadovoljava zahtjeve jer više diskova povećava vjerojatnost kvara

dakle: 20 TB / 5 TB => 4 diska za kapacite => 6 diskova u RAID 6

Neki su naveli RAID1, jer i tamo se mogu pokvariti dva diska (ili više) a da sustav radi. Međutim ne mogu bilo koja dva diska! Kod RAID6 mogu bilo koja dva diska biti neispravna a da sustav i dalje radi.

12. (8) Problem sličan problemu pet filozofa (*ponavljaj {misli + jedi}*) pojavljuje se u „Restoranu na kraju svemira“. Tamo za istim stolom sjede osobe s različitim brojem ruku. Pretpostaviti da se početno na sredini stola nalazi *N* vilica i da svaka osoba koja jede mora barem u svakoj drugoj ruci imati jednu vilicu kad jede. Npr. osoba koja ima pet ruku treba minimalno dvije vilice. Osoba će čekati ako na stolu nema minimalan potreban broj vilica. U protivnom uzima vilice sa stola (minimalan broj ili ako ih ima više i više, dok svaka ruka nema po jednu) te jede. Nakon što se najede osoba opere vilice i vraća ih na sredinu stola. Simulirati osobe dretvama i sinkronizirati ih monitorima. Napisati početne vrijednosti korištenih varijabli. Nije potrebno rješavati problem izgladnjivanja~~, ali navesti ako postoji u rješenju te ako da scenarij njegove pojave~~.

```
br_vilica = N
monitor M
red uvjeta R

dretva osoba(br_ruku) {
   ponavljaj {
      misli

      Uđi_u_monitor(M)
      dok je (br_vilica < br_ruku / 2) //pretpostavlja se cjelobrojno dijeljenje
         Čekaj_u_redu_uvjeta(R, M)
      uzeo = min(br_vilica, br_ruku) //"uzeo" = lokalna varijabla
      br_vilica -= uzeo
      Izađi_iz_monitora(M)

      jedi

      Uđi_u_monitor(M)
      br_vilica += uzeo
      Oslobodi_sve_iz_reda_uvjeta(R)
      Izađi_iz_monitora(M)
   }
}
```

Osoba koja treba više vilica može biti izgladnjivana ako u sustav dolaze osobe koje trebaju manje vilica...