Dodatni zadaci/primjeri iz *Operacijskih sustava* (uz teoretska!)

(crno=zadano, plavo=rješenje)

1. Slojevita izgradnja

ZAD: 1.1. Prikazati primjere korištenja sučelja u računalnom sustavu

Korisnik koristi vanjske naprave te teoretski komunicira sa sklopovljem. Međutim, na višoj razini te akcije idu prema programima i OS-u.

Korisnik → programi

- unos parametara
- odabir opcija (iz menija, pritisak na gumb, ...)
- interakcija tipkovnicom, mišem, zaslonom osjetljivim na dodir

Korisnik → OS

- pokretanje programa (iz konzole zadavanjem naredbe, mišem, ...)
- zaustavljanje programa
- upravljanje programima (prioritet, ...)

Korisnik → sklopovlje

- upali računalo, ugasi, resetiraj, korištenje vanjskih naprava: zaslon, tipkovnica, miš, ...

Programi → OS

- korištenje naprava (ispiši znakove, učitaj)
- korištenje podsustava (rad s datotekama, mrežnim podsustavom, ...)
- sinkronizacija i komunikacija među dretvama i procesima

Programi → sklopovlje

- izvođenje instrukcija programa

OS → sklopovlje

- izvođenje instrukcija jezgre
- upravljanje procesorom (upravljačkim registrima: registar stanja, ...)
- upravljanje napravama: slanje i čitanje podataka, postavljanje preko upravljačkih registara naprava, obrada prekida

2. Prekidi

ZAD: 2.1. U nekom sustavu brzina sabirnice i procesora je 200 MHz (sab. ciklus traje 5 ns). Kontekst dretve čini 20 registara. Određivanje uzroka prekida zahtijeva izvođenje 60 instrukcija prije skoka na proceduru za obradu. Koliko traju kućanski poslovi (spremanje konteksta, određivanje uzroka prekida, obnova konteksta) po svakom prekidu?

spremanje i obnova konteksta: 20 sab. ciklusa * 2 = 40 sab. ciklusa
određivanje uzroka prekida 60 instr. = 60 sab. ciklusa
ukupno= 40+60 = 100 instrukcija => 100 * 1 / 200 000 000 = 1 / 2 000 000 = 0,5 μs
= **500 ns**

ZAD: 2.2. Prekid P1 javlja se svakih 100 ms. Prekid P2 javlja se svakih 150 ms. P1 i P2 nisu sinkronizirani. Obrada od P1 traje između 5 i 15 ms, a obrada P2 između 15 i 20 ms. Prekidi se obrađuju po redu prispijeća, uz zabranjeno prekidanje za vrijeme obrade. Zanemariti vremena kućanskih poslova (prihvat prekida, spremanje i obnova konteksta, ...).

a) Koliko najviše P1 mora čekati na početak obrade?
20 ms
b) Koliko najviše P2 mora čekati na početak obrade?
15 ms
c) Koji postotak procesorskog vremena sigurno uvijek ostaje za druge potrebe?

svakih 300 ms (prvi višekratnik 100 i 150):

- P1 se javlja 3 puta po 15 ms (max) = 3 * 15 = 45 ms
- P2 se javlja 2 puta po 20 ms (max) = 2 * 20 = 40 ms
- 300 – 45 – 40 = 215 ms; 215/300 = **71,66̄ %**.

*Ponoviti c) ukoliko kućanski poslovi traju 1 μs.

uz svaki prekid dodatno uzeti i trošak kućanskih poslova:
5 prekida * 1 μs = 5 μs => (215-0,005)/300 = 71,665 %

**Ponoviti c) ukoliko kućanski poslovi traju 1 μs te ukoliko P1 prekida obradu prekida P2 (ali se P2 kasnije nastavlja obrađivati, nakon obrade P1).

u 3 pojavljivanja P1 može jednom ili dva puta prekinuti P2, međutim, to ne mijenja broj kućanskih poslova!

3. Jezgra, semafori, sinkronizacija

ZAD: 3.1. Napisati jezgrine funkcije ČekajSem(i) i PostaviSem(i).

```
j-funkcija CekajSem(id)
{
	ako (Sem[id].vrijednost > 0) {
		Sem[id].vrijednost--;
	}
	inače {
		Stavi_u_red(Aktivna_D, Sem[id].red);
		Aktivna_D = UzmiPrvu(Pripravne_D);
	}
}

j-funkcija PostaviSem(id)
{
	ako (Sem[id].red neprazan) {
		Stavi_u_red(Aktivna_D, Pripravne_D);
		prva = UzmiPrvu(Sem[id].red);
		Stavi_u_red(prva, Pripravne_D);
		Aktivna_D = UzmiPrvu(Pripravne_D);
	}
	inače {
		Sem[id].vrijednost++;
	}
}
```

ZAD: 3.2. U promatranom trenutku stanje sustava je slijedeće: dretva 1 je aktivna; dretve 2 i 3 su u redu semafora 1 te dretve 4 i 5 su u redu pripravnih dretvi. Ako tada dretva 1 pozove jezgrinu funkciju PostaviSem(1), kako će izgledati struktura podataka jezgre NAKON poziva? Svi redovi organizirani su po redu prispijeća (FIFO).

prije: Aktivna = 1 Pripravne = 4,5 SEM[1] = 2,3

akcija: PostaviSem(1)

(prema opisu funkcije: najprije 1 ide u pripravne, potom prva iz reda semafora (2) u red pripravnih te na kraju prva iz reda pripravnih (4) u aktivnu)

nakon: Aktivna = 4 Pripravne = 5,1,2 SEM[1] = 3

Što ako su složeni prema prioritetu (indeks dretve označava njen prioritet - manji broj je veći prioritet?

prije: Aktivna = 1 Pripravne = 4,5 SEM[1] = 2,3

akcija: PostaviSem(1)

(prema opisu funkcije: najprije 1 ide u pripravne, potom prva iz reda semafora (2) u red pripravnih te na kraju prva iz reda pripravnih (1) u aktivnu)

nakon: Aktivna = 1 Pripravne = 2,4,5 SEM[1] = 3

Što ako se pozove ČekajBSem(1) umjesto PostaviBSem(1)?

prije: Aktivna = 1 Pripravne = 4,5 SEM[1] = 2,3

akcija: ČekajSem(1)

(prema opisu funkcije: najprije 1 ide u red red semafora, te potom prva iz reda pripravnih (4) u aktivnu; semafor je sigurno neprolazan obzirom da ima dretvi u njegovu redu!)

nakon: Aktivna = 4 Pripravne = 5 SEM[1] = 2,3,1

ZAD: 3.3. Tri dretve sudjeluju u obavljanju zajedničkog posla na način da dretva D1 dohvaća podatke te podatke tipa A1 prosljeđuje dretvi D2 preko varijable M1 (kapaciteta jedne poruke), a sve ostale podatke prosljeđuje dretvi D3 preko međuspremnika M2 kapaciteta 5 poruka. Napisati funkcije `stavi(poruka, tip)` i `uzmi(tip)`, tj. nadopuniti slijedeći kod.

```
stavi (poruka, tip) {
   ako je ( tip == A1 ) {
      ČekajSem(1);
      M1 = poruka;
      PostaviSem(2);
   } inače {
      ČekajSem(3);
      M2[UL] = poruka;
      UL = (UL+1) mod 5;
      PostaviSem(4);
   }
}
```

```
uzmi (tip) {//vraća „nporuka"
   ako je ( tip == A1 ) {
      ČekajSem(2);
      nporuka = M1;
      PostaviSem(1);
   } inače {
      ČekajSem(4);
      nporuka = M2[IZ];
      IZ = (IZ+1) mod 5;
      PostaviSem(3);
   }
   vrati (nporuka);
}
```

Početne vrijednosti semafora: Sem[1].v=1, Sem[2].v=0, Sem[3].v=5, Sem[4].v=0
Početne vrijednosti varijabli: UL = IZ = 0

ZAD: 3.4. U sustavu se nalaze dvije dretve, dretva *Ping* i dretva *Pong*. Nadopuniti slijedeće funkcije potrebnim jezgrinim funkcijama, ako dretve trebaju naizmjence ispisivati „Ping“ i „Pong“ na zaslon. Sinkronizaciju obaviti korištenjem semafora. Napisati početne vrijednosti semafora.

```
Dretva Ping() {
   ...
   ponavljaj {
      ...
      Ispiši(„Ping“);
      ...
   } do ZAUVIJEK;
   ...
}
```

```
Dretva Pong() {
   ...
   ponavljaj {
      ...
      Ispiši(„Pong“);
      ...
   } do ZAUVIJEK;
   ...
}
```

```
Dretva Ping() {
   ponavljaj {
      ČekajSem(1);
      Ispiši(„Ping“);
      PostaviSem(2);
   } do ZAUVIJEK;
}
```

```
Dretva Pong() {
   ponavljaj {
      ČekajSem(2);
      Ispiši(„Pong“);
      PostaviSem(1);
   } do ZAUVIJEK;
}
```

*Dretva *Ping* treba ispisati N puta „Ping” te potom dretva *Pong* M puta „Pong” i opet ispočetka.

```
Dretva Ping() {
   ponavljaj {
      ČekajSem(1);
      za i=1 do N {
         Ispiši(„Ping“);
      }
      PostaviSem(2);
   } do ZAUVIJEK;
}
```

```
Dretva Pong() {
   ponavljaj {
      ČekajSem(2);
      za i=1 do M {
         Ispiši(„Pong“);
      }
      PostaviSem(1);
   } do ZAUVIJEK;
}
```

*Dretva *Ping* treba ispisati N puta „Ping” te potom dretva *Pong* M puta „Pong” i opet ispočetka. Kôd dretve *Ping* je zadan i ne može se promijeniti, tj. sve treba ugraditi u dretvu *Pong*.

```
Dretva Ping() {
   ponavljaj {
      ČekajSem(1);
      Ispiši(„Ping“);
      PostaviSem(2);
   } do ZAUVIJEK;
}
```

```
Dretva Pong() {
   ing = 0;
   ponavljaj {
      ČekajSem(2);
      ing++;
      ako je ( ing == N ) {
         ing = 0;
         za i=1 do M {
            Ispiši(„Pong“);
         }
      }
      PostaviSem(1);
   } do ZAUVIJEK;
}
```

4. Upravljanje spremnikom

ZAD: 4.1. Strategije izbacivanja

U sustavu sa straničenjem program veličine 400 riječi (1-400) generira slijed adresa:

```
23, 47, 333, 81, 105, 1, 400, 157, 30, 209, 289, 149, 360
```

Program ima na raspolaganju 200 riječi primarne memorije. Napisati niz referenciranja stranica veličine 50 riječi. Koliki je postotak promašaja za sve tri strategije izbacivanja stranica.

Rješenje:

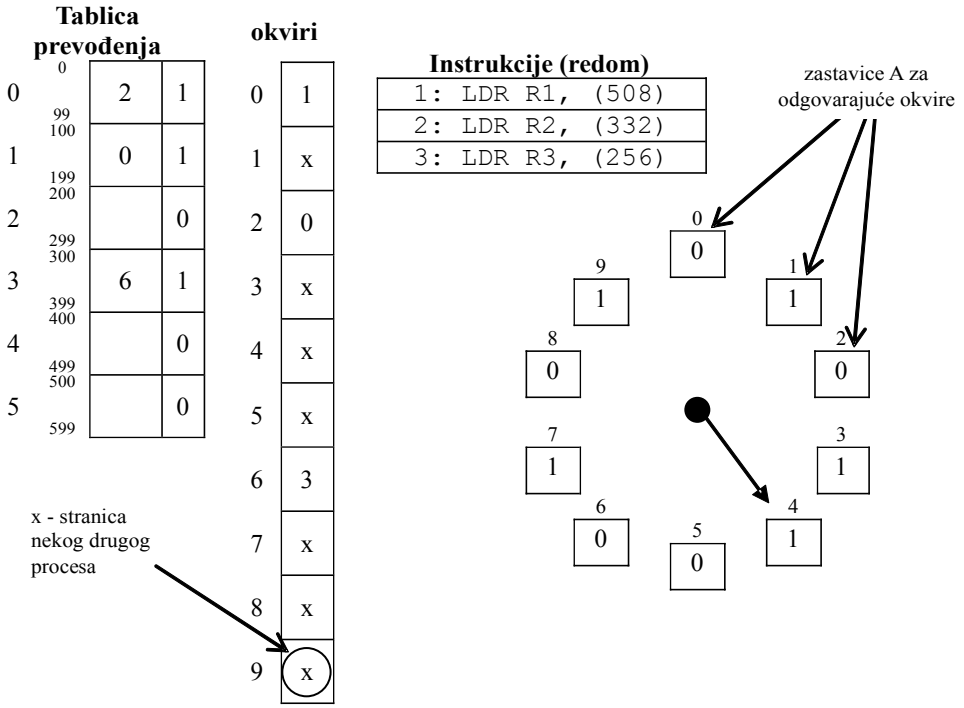
adresa	23	47	333	81	105	1	400	157	30	209	289	149	360
stranica ili zahtjev	1	1	7	2	3	1	8	4	1	5	6	3	8
a) FIFO 11/13*100%	**1** **-** **-** **-**	| | | |	**1** **7** **-** **-**	**1** **7** **2** **-**	**1** **7** **2** **3**	| | | |	**8** **7** **2** **3**	**8** **4** **2** **3**	**8** **4** **1** **3**	**8** **4** **1** **5**	**6** **4** **1** **5**	**6** **3** **1** **5**	**6** **3** **8** **5**
b) LRU 10/13*100%	**1** **-** **-** **-**	| | | |	**1** **7** **-** **-**	**1** **7** **2** **-**	**1** **7** **2** **3**	| | | |	**1** **8** **2** **3**	**1** **8** **4** **3**	| | | |	**1** **8** **4** **5**	**1** **6** **4** **5**	**1** **6** **3** **5**	**8** **6** **3** **5**
c) OPT 8/13*100%	**1** **-** **-** **-**	| | | |	**1** **7** **-** **-**	**1** **7** **2** **-**	**1** **7** **2** **3**	| | | |	**1** **8** **2** **3**	**1** **8** **4** **3**	| | | |	**5** **8** **4** **3**	**5** **8** **6** **3**	| | | |	| | | |

Dodatno: Prikažite trenutni izgled tablice prevođenja programa na kraju primjene OPT strategije (u tablicu uključiti bit prisutnosti)

Broj stranice (redak tablice)	Broj okvira	Bit prisutnosti
1	1	0
2	3	0
3	4	1
4	3	0
5	1	1
6	3	1
7	2	0
8	2	1

ZAD: 4.2. Satni algoritam

Za neki sustav koji koristi straničenje, zadana je tablica prevođenja za jedan proces, stanje okvira sustava, stanje kazaljke i zastavica A (za upravljanje metodom satnog algoritma). Ako zadani proces treba napraviti zadane instrukcije, kako će se sustav mijenjati? Neka se instrukcije nalaze u 0. stranici.

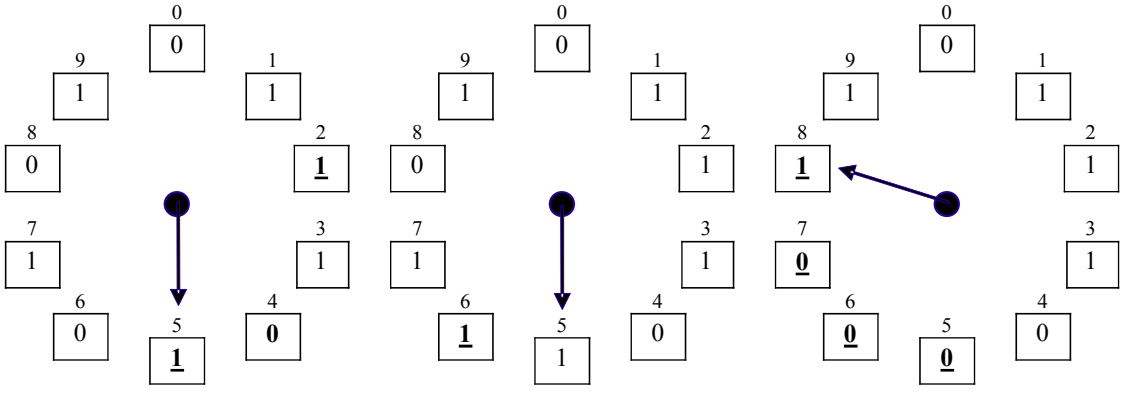


Rješenje:

Dohvat prve instrukcije izazvat će pogodak (0. stranica je u 2. okviru). Izvođenjem prve instrukcije dogoditi će se prekid zbog promašaja podataka (traži se stranica 5 programa). Zastavica četvrtog okvira A(4) postaviti će se u 0 te će se kazaljka pomaknuti. Zastavica A(5) je 0 te će se taj okvir osloboditi i u njega staviti 5. stranica procesa. Tada se može obaviti prva instrukcija. Njenim izvođenjem (čitanjem iz 5. okvira) postavlja se zastavica A(5) u 1 (sl. a)).

Druga instrukcija traži 3. stranicu koja se nalazi u okviru 6. Njenim izvođenjem (čitanjem podatka iz 6. okvira) postaviti će se zastavica A(6) u 1 (sl. b)) (kazaljka se ne miče).

Treća instrukcija traži podatak iz 2. stranice koja nije u radnom spremniku, pa će se kazaljka pomaknuti, najprije na 6. mjesto, pa na 7. (pritom postavlja A(5), A(6) i A(7) u nulu) i tek na 8. pronalazi A(8)=0, izbacuje stranicu koja se tu nalazi i učitava stranicu 2 procesa. Nakon toga može se izvesti instrukcija 3. Izvođenjem 3. instrukcije postavlja se zastavica A(8) u 1 (sl. c)).



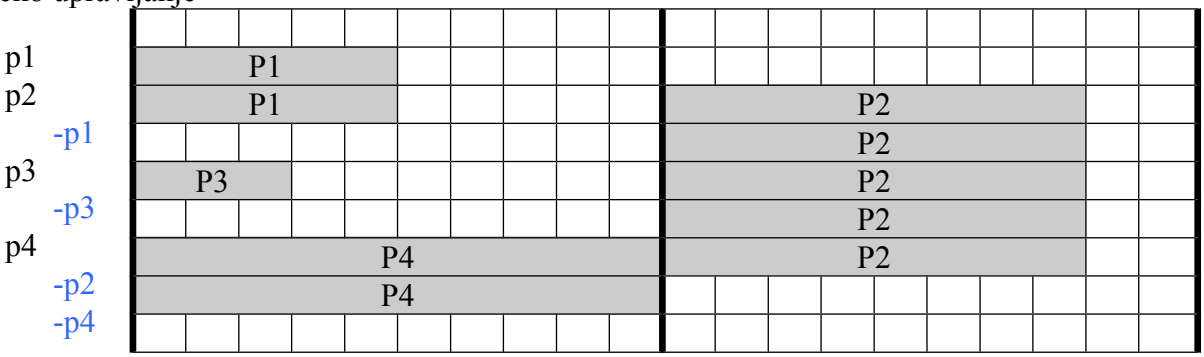
a) nakon 1. instr. b) nakon 2. instr. c) nakon 3. instr.

ZAD: 4.3. Usporedba upravljanja spremnikom

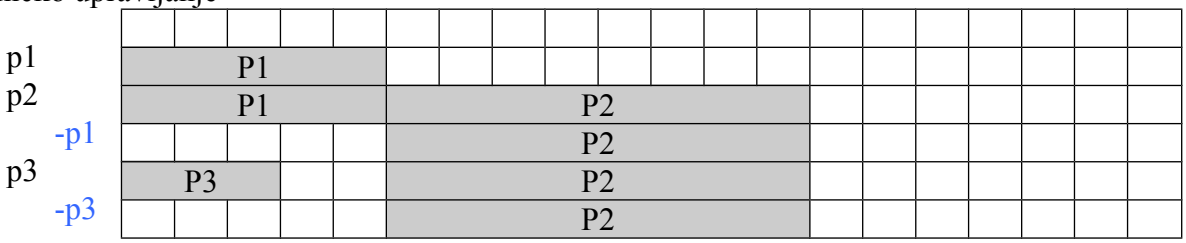
U nekom sustavu trebaju se obaviti četiri programa: P1, P2, P3 i P4 koji su već pripremljeni na pomoćnom spremniku i zauzimaju redom 5 MB, 8 MB, 3 MB, 10 MB. Događaji pokretanja i završetka programa znani su unaprijed i mogu se iskazati slijedećim nizom događaja: P1 pokrenut; P2 pokrenut; P1 završava; P3 pokrenut; P3 završava; P4 pokrenut; P2 završava; P4 završava. Sustav na raspolaganju ima 20 MB memorije rezervirane za korisničke programe. Prikazati stanje radnog spremnika ako se koriste metode upravljanja spremnikom:

a) statičko upravljanje s veličinom segmenta od 10 MB,
b) dinamičko upravljanje,
c) straničenje, uz veličinu stranice od 1 MB.

a) statičko upravljanje



b) dinamičko upravljanje



P4 se ne može smjestiti u memoriju u ovom trenutku! (P3 ide u najmanju rupu!)

c) straničenje

P1	P1	P1	P1	P1															
P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P2	P2							
P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P2	P2							
P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P2	P2	P3	P3	P3				
P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P2	P2	P3	P3	P3				
P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P2	P2	P3	P3	P3	P4	P4	P4	P4
*	*	*	*	*	P2	P2	P2	P2	P2	P2	P2	P2	*	*	*	P4	P4	P4	P4
P4	P4	P4	P4	P4	P2	P2	P2	P2	P2	P2	P2	P2	P4			P4	P4	P4	P4
P4	P4	P4	P4	P4	P2	P2	P2	P2	P2	P2	P2	P2	P4			P4	P4	P4	P4
P4	P4	P4	P4	P4	P2	P2	P2	P2	P2	P2	P2	P2	P4			P4	P4	P4	P4

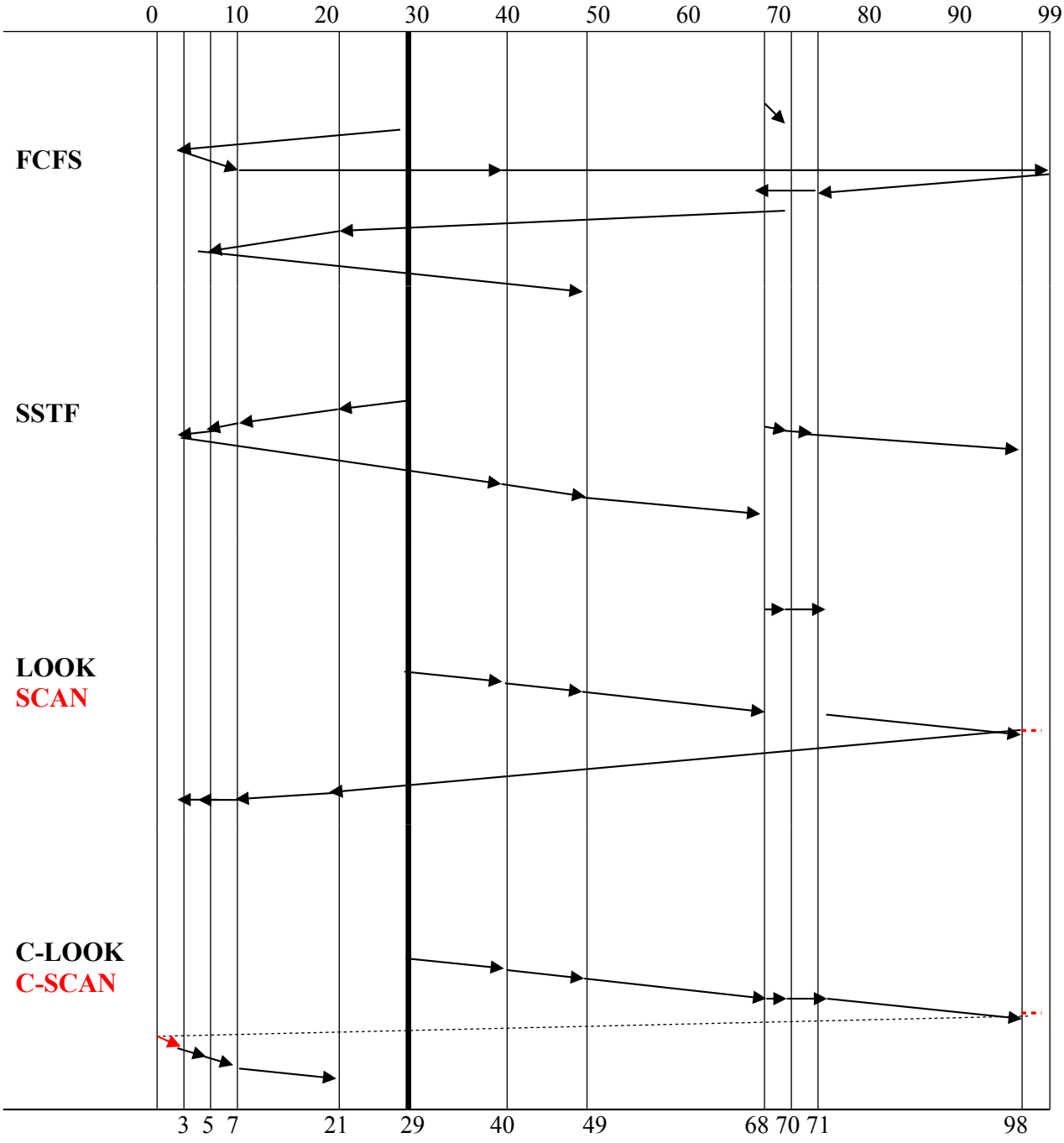
* - nije bilo mjesta pa je trebalo obrisati memoriju prije dodjeljivanja

5. Datotečni podsustav

ZAD: 5.1. Disk s pokretnim glavama ima 100 staza (0 - 99). Neka se glava trenutno nalazi na stazi 29, s tim da je prije bila na stazi 8. Zahtjevi za pristup pojedinim stazama svrstani po redu prispjeća su: **3, 7, 40, 98, 71, 68, 70, 21, 5 i 49.**

a) Koliki je ukupan pomak glave pri izvršenju tih zahtjeva za strategije posluživanja: redom prispjeća (FCFS), s najkraćim vremenom premještaja glave (SSTF), pregledavanje (LOOK i SCAN) i kružno pregledavanje (C-LOOK i C-SCAN)?
b) Opisati svaku od navedenih strategija!
c) Kod kojih može doći do izgladnjivanja (beskonačnog odgađanja posluživanja nekih zahtjeva zbog posluživanja novo pristiglih zahtjeva)?
d) Grafički prikazati kretanje glave prilikom obrade zahtjeva za sve strategije!

Rješenje:



ZAD: 5.2. Disk ima 100 sektora po stazi veličine 1kB, 2000 staza, 4 ploče i vrti se brzinom 7200 okretaja u minuti. Podaci su zapisani na obje strane ploče (ukupno 8 površina). Upravljački sklop pročita jednu cijelu stazu u interni spremnik, a zatim je prenosi u glavni spremnik. Prijenos u glavni spremnik odvija se brzinom od 200 Mbit/s, a za to vrijeme sklop ne može čitati s diska.

a) Koliki je kapacitet tog diska?
b) Koliko prosječno traje prebacivanje **kompaktno** smještene datoteke veličine 1350 KB ako je vrijeme postavljanja 10 ms i vrijeme premještanja sa staze na stazu 1 ms?

Rješenje:
a)
Kapacitet diska = 1kB * 100 * 2000 * 4 * 2 = 1.526 GB

b)
Kapacitet jedne staze = 100 * 1KB = 100 KB
Datoteka od 1350 KB je na 1350/100 = 14 staza. **Kompaktno** znači jedan cijeli cilindar (8 staza) te dio drugog (5 i pol staze)!

Prijenos datoteke =

- pronalaženje početka datoteke = trajanje postavljanja glave + rotacijsko kašnjenje

- (i) trajanje postavljanja glave = 10 ms
- (ii) rotacijsko kašnjenje = $T_R/2$ = 4,16 ms
- (iii) čitanje staze (T_R) = 8,33 ms
- (iv) prijenos podataka (100kB) = 100 KB / 200 Mbit/s = 4,096 ms
- ponovi (ii)+(iii)+(iv) 10 puta (na 1. cilindru) = 10 * (4,16+8,33 ms + 4,096 ms)

- pri zadnjem prijenosu podataka (10.), moglo se paralelno pomicati glavu na susjednu stazu
- obzirom da je premještanje glave = 1ms, glava već je na susjednoj stazi kad je prijenos podataka gotov

- ponovi (ii)+(iii)+(iv) 6 puta (na 2. cilindru) = 6 * (4,16+8,33 ms + 4,096 ms)

- komentar: zadnji dio = 50 KB nalazi se na 50 sektora = 1/2 staze, ali ipak čitamo cijelu i cijelu ju prenosimo u radni spremnik (tako je zadano u zadatku!)

= 10 + 10 * (4,16+8,33 ms + 4,096 ms) + 6 * (4,16+8,33 ms + 4,096 ms) = **275,376 ms**

Neka je zadana datoteka.

Datoteka – logička organizacija

indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sadržaj	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Na disku:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18 **C**	19 **D**	20 **E**	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37 **A**	38 **B**	39	40
41	42	43	44	45	46	47	48
49	50	51 **K**	52 **L**	53 **M**	54 **N**	55 **O**	56
57 **F**	58 **G**	59 **H**	60 **I**	61 **J**	62	63	64

ZAD: 5.3. Kako izgleda struktura podataka u NTFS tablici koja opisuje smještaj te datoteke?

Opis smještaja:

VCN	LCN	Broj nakupina
1	37	2
3	18	3
6	57	5
11	51	5

ZAD: 5.4. Kako izgleda struktura podataka u UNIX inode tablici (uz pretpostavku 10+3 kazaljke) koja opisuje smještaj te datoteke?

37	38	18	19	20	57	58	59	60	61	(x)	(-)	(-)

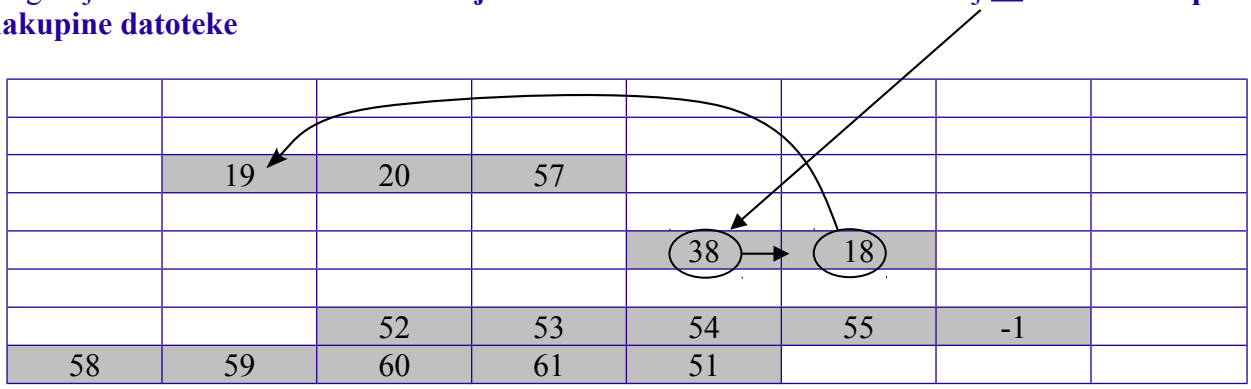
(x) je kazaljka na blok s kazaljkama:

51	52	53	54	55	(neiskorištene kazaljke)

ZAD: 5.5. Kako izgleda struktura podataka u FAT tablici koja opisuje smještaj te datoteke?

FAT koristi *tablicu direktorija* (engl. *directory table*) i *tablici zauzeća* (engl. *file allocation table*)
U *tablici direktorija* nalazi se opisnik datoteke sa imenom i ostalim atributima te adresu prvog bloka datoteke (u kojem se bloku particije nalazi).
Tablica zauzeća je zajednička za sve datoteke. Tablica ima onoliko zapisa koliko ima blokova na particiji. U svakom zapisu (broju, 32-bitovni broj za FAT32) pokazuje se na idući blok datoteke, ili -1 ako je ovo zadnji blok, ili 0 ako je blok slobodan (ne koristi se).

Za gornju datoteku u **tablici direktorija** za zadanu datoteku nalazio bi se broj **37** kao oznaka **prve nakupine datoteke**



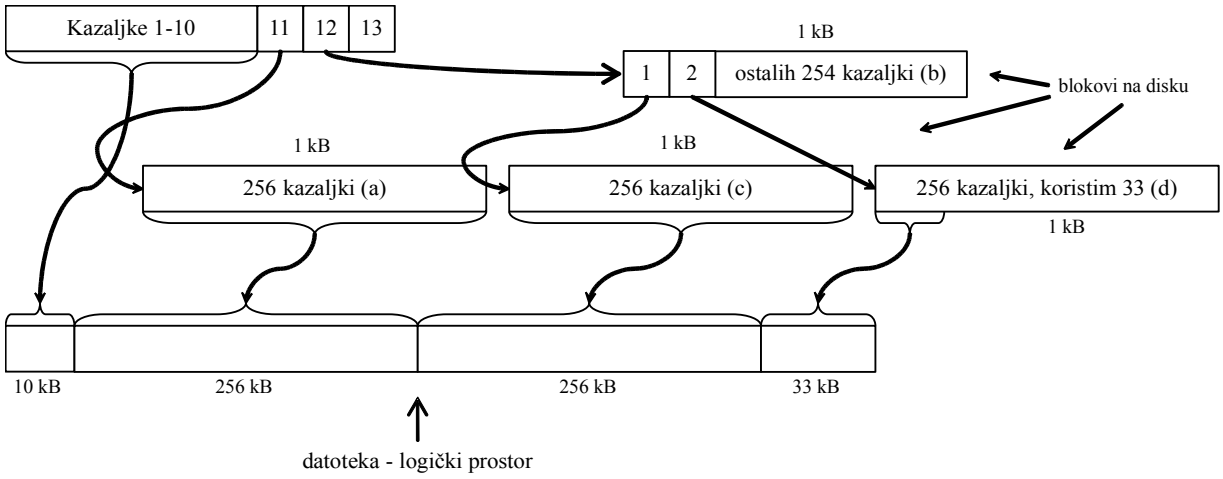
ZAD: 5.6. Ako je na UNIX datotečnom sustavu pohranjena datoteka veličine 555 kB koliko memorijskog prostora zauzimaju kazaljke za tu datoteku u tablici datotečnog sustava? Skicirajte organizaciju tih kazaljki. Veličina bloka je 1024 okteta (1kB), a veličina kazaljke 32 bita.

Opisnik 10+3 kazaljke (*paralelno crtaj graf*)

Prvih 10 opisuje prvih 10 * 1024 okteta = 10 kB, ostaje 555 – 10 = 545 kB

11. kazaljka pokazuje na blok sa 1024/(32/8) = 256 kazaljki => 256 kB, ostaje 545-256 = 289 kB

12. kazaljka pokazuje na blok koji može imat 256 kazaljki na blokove s kazaljkama
S obzirom da nam je ostalo 289 kB, a svaki blok može imati 256 kazaljki, tj. opisati 256 kB, dovoljna su nam dva takva bloka. Tj. od bloka na koji pokazuje 12. kazaljka koristimo samo prve dvije kazaljke, a blok druge kazaljke ne koristimo u cijelosti.
[12.kaz] → [kaz1, kaz2, …. kaz256];
[kaz1] → [sekt1, sekt2, … sekt256]; pokriva 256 kB, ostaje 289-256 = 33
[kaz2] → [sekt1, sekt2, … sekt33, 0, 0, …., 0]; pokriva 33 kB



Kazaljke zauzimaju (bez onih u opisniku) a+b+c+d = 4 bloka, tj. 4 kB (iako neke kazaljke u blokovima 'b' i 'd' nisu iskorištene).

Dodatno:
Najveća datoteka?
10 KB + 256 KB + 256*256 KB + 256*256*256 KB = 16843018 KB ≈ 16448 MB ≈ 16 GB
Blokova za kazaljke: 1 + (1 + 256) + (1 + 256 * (1 + 256)) = 66051 (KB) ≈ 64,5 MB
(11) (12) (13)
Datoteka od 3 GB?
1.-10. → 10 KB, ostaje 3 GB – 10 KB = 3*1024*1024 – 10 = 3145718 KB
11. → 256 KB, ostaje 3145718 – 256 = 3145462 KB (1 blok za kazaljke)
12. → 256*256 = 65536 KB, ostaje 3145462 – 65536 = 3079926 KB (1+256 blokova za kazaljke)

13. → 3079926 KB = 46*65536 + 65270 (1 blok za početne kazaljke, samo 47 od 256 se koristi)
→ prvih 46 kazaljki će pokazivati na blokove potpuno iskorištene (46*(1+256) blokova za k.)
→ 47 kazaljka: 65270 = 254 * 256 + 246 (1+255 blokova za kazaljke)

Ukupno blokova za kazaljke: 1 + (1 + 256) + (1 + 46 * (1 + 256) + 1 * (1 + 255)) = 12337
„Višak kazaljki“? Koliko treba minimalno ? 3 GB = 3 * 1024 * 1024 blokova za podatke; za svaki blok treba biti adresa=4 bajta (32 bita) → 3 * 1024 * 1024 * 4 B = 12 MB; 12337 blokova = 12 MB + 49 KB