OPERACIJSKI SUSTAVI

Ponavljanje I1+I2+I3

Operacijski sustavi

- Ishod 1: Objasniti rad prekidnog sustava na modelu jednostavnog računala
  - Grafovi prekida
  - Kako nastaje prekid i kako se obrađuje
- Ishod 2: Objasniti pojam procesa na računalu
  - Što je proces
  - Vrste procesa
  - Naredba fork()
- Ishod3: Objasnite koncept dretvi na računalu i kako im procesor dodjeljuje vrijeme
  - Što je kritični odsječak
  - Algoritmi/strategije
  - Grafovi stanja CPU i spremnika kod algoritama

I1. Prekidi/Interrupts

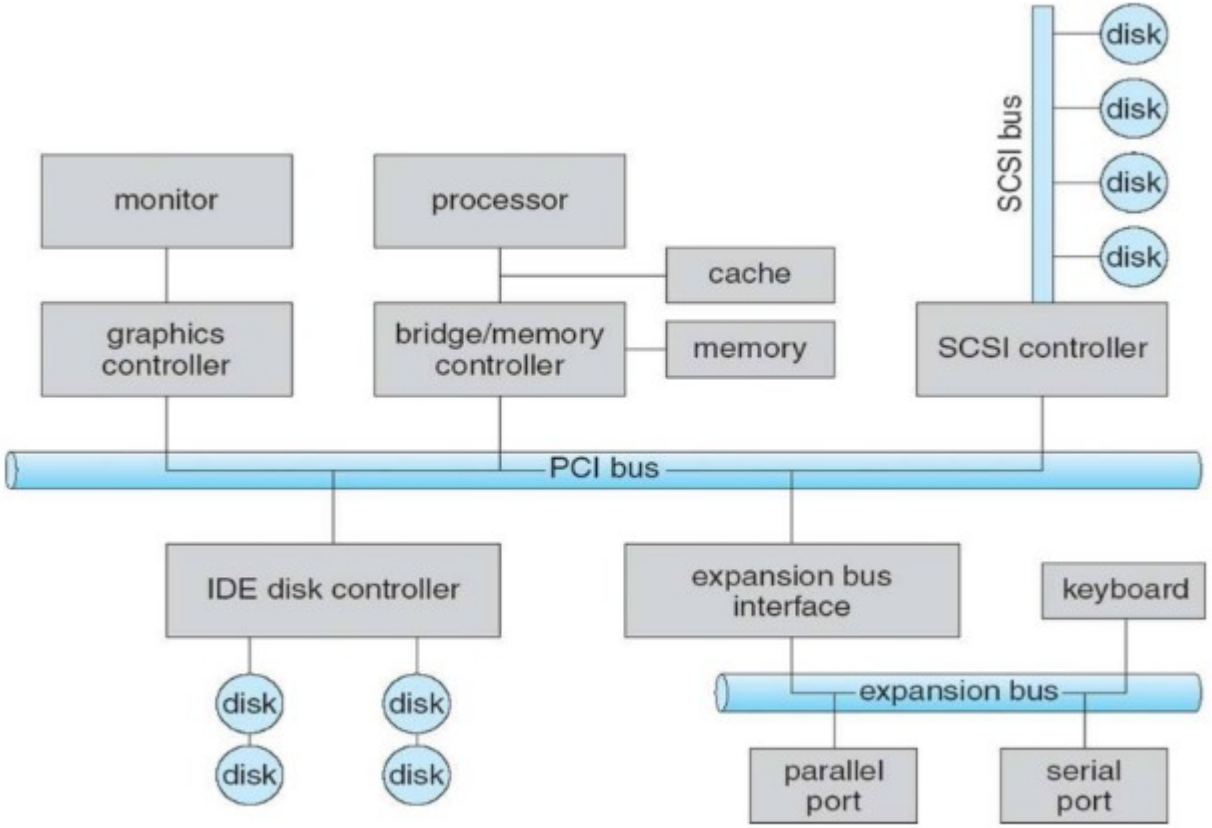
Što je Prekid?

- Signal koji se šalje od računalne komponente ili programa prema Operacijskom sustavu
- Uzrokuje da OS privremeno zaustavi sa svojim redovitim procesima i počinje obrađivati zahtjev za prekid (*interrupt handler*)
- Prekidi imaju prioritet
  - Prekidna instrukcija može uzrokovati da se neki trenutni procesi prebrišu ili utječe na njihovo kašnjenje

English: Interrupt (Hardware or Software), TRAP (CPU)

Računalo

- Svaki sklop ima svoj IRQ
- Zahtjev za prekidom (ili IRQ) je hardverski signal koji se šalje procesoru koji privremeno zaustavlja pokrenuti program i dopušta da se umjesto toga pokrene poseban proces, rukovatelj prekida – Interrupt handler.

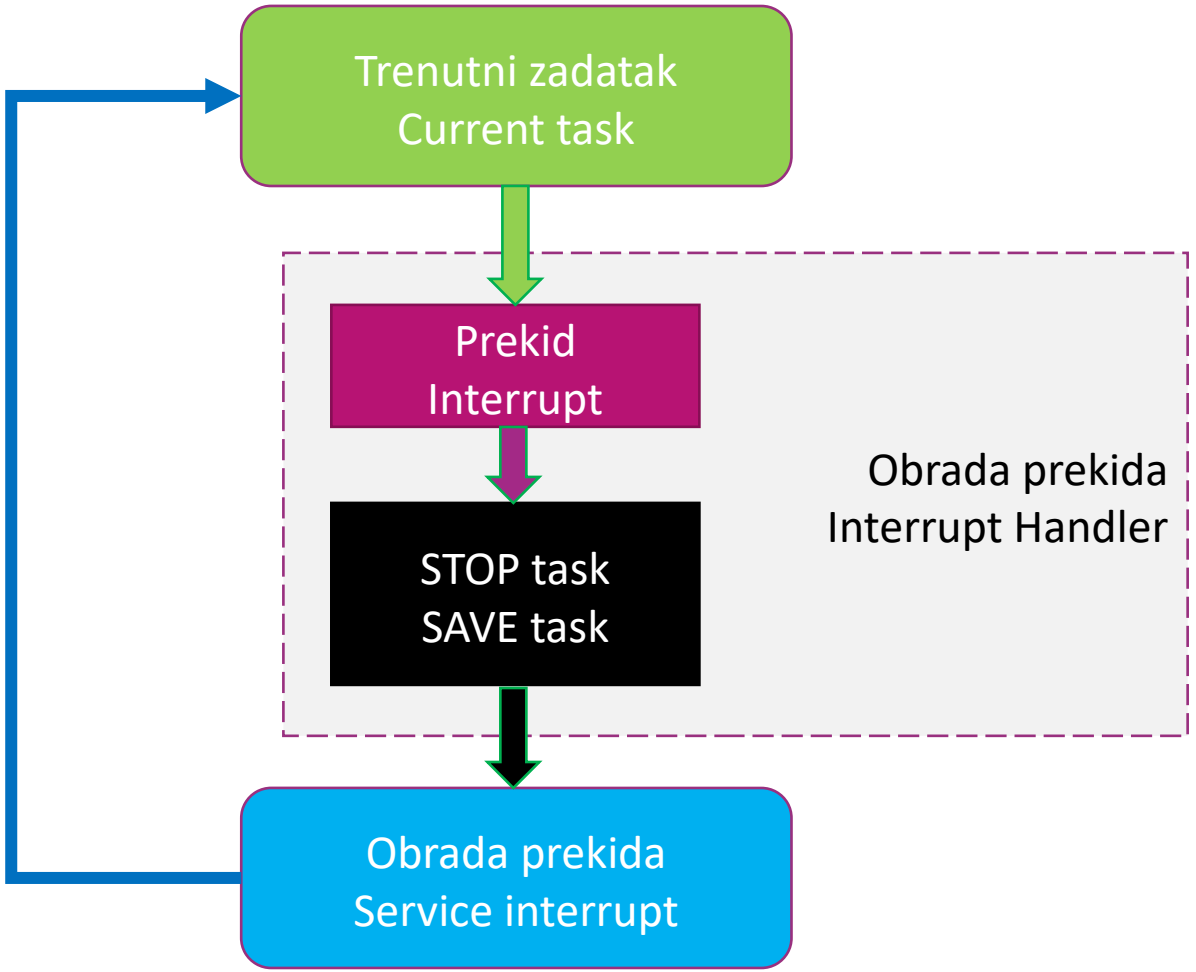


Važnost prekida

- Mogućnost da aplikacije prekinu jednu drugu daje iluziju multitaskinga
- Korisniku daju bolju kontrolu na računalu (bez prekida trebamo čekati da aplikacija završi)
- Traju jako kratko (0,1% - 2% CPU)*
  - Ovisno o aplikaciji mogu doseći vrijednosti od 3%-7%
  - (u pravilu) Ako traju više od 5% - hardware greška!

* U pravilu traju od 0,1% do 0,2%, kod normalnog rada, bez aplikacije koja uzrokuje „redovne” prekide

Obrada prekida



Zadatak 1.1.

Nacrtajte obradu zadanog prekidnog sustava ako računalo u vremenskom periodu 22t mora obaviti 5 prekida gdje najviši prioritet ima P5, a najmanji P1.

Trajanje i pojavljivanje prekida:

P5 - pojavljivanje - 4t, trajanje - 2t

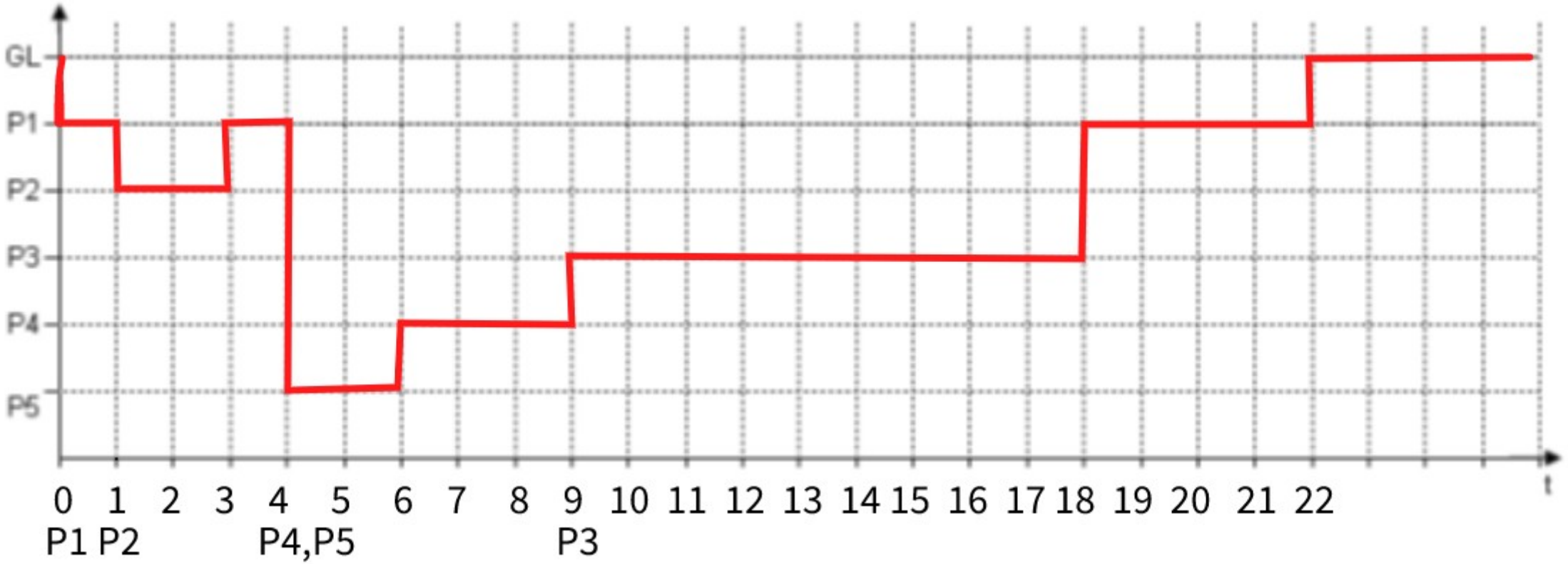
P4 - pojavljivanje - 4t, trajanje - 3t

P3 - pojavljivanje - 9t, trajanje - 9t

P2 - pojavljivanje - 1t, trajanje - 2t

P1 - pojavljivanje - 0t, trajanje - 6t

Rješenje 1.1.


GL
P1
P2
P3
P4
P5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
t
P1 P2
P4,P5
P3

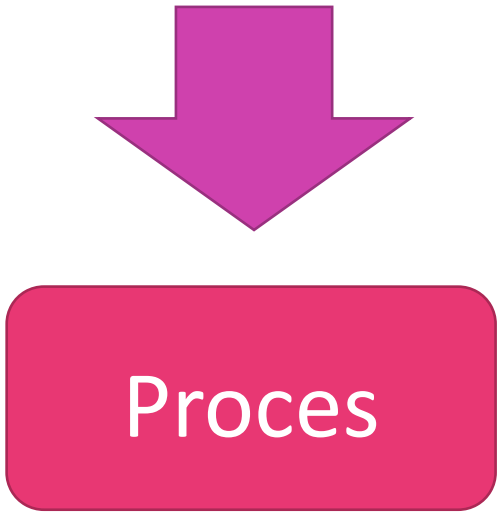
I2. Procesi

Osnovni pojmovi: Proces

- Proces
  - Instanca programa koja se izvodi
  - Jedan program može imati više instanci (npr. dva puta se otvori)
  - OS se brine da procesi imaju odvojene resurse (nemoguće je pristupiti varijabli drugog procesa iz prvog procesa)

Kako nastaje proces

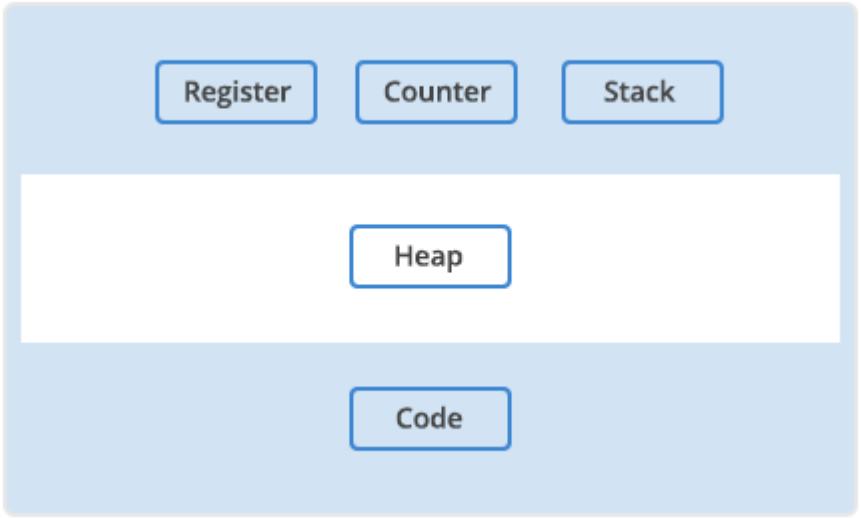
- Program u binarnom obliku je učitan u radnu memoriju
- Program rezervira dodatnu memoriju za svoje varijable
- Program rezervira razne resurse operativnog sustava



- Operativni sustav je mozak za alociranje svih potrebnih resursa

Resursi procesa

- Registri (*Register*)
  - Instrukcije, pohrana adresa…
- Brojači (*Counter*)
  - Vodi brigu koja je trenutna i sljedeća instrukcija na izvršavanju
- Stog (*Stack*)
  - služi za pohranu niza istovrsnih elemenata (omogućava upis i ispis po principu "zadnji koji ulazi - prvi izlazi„ - FIFO)
- Dinamički alocirana memorija (*Heap*)



Detaljnije na predmetu Građa računala

Više procesa

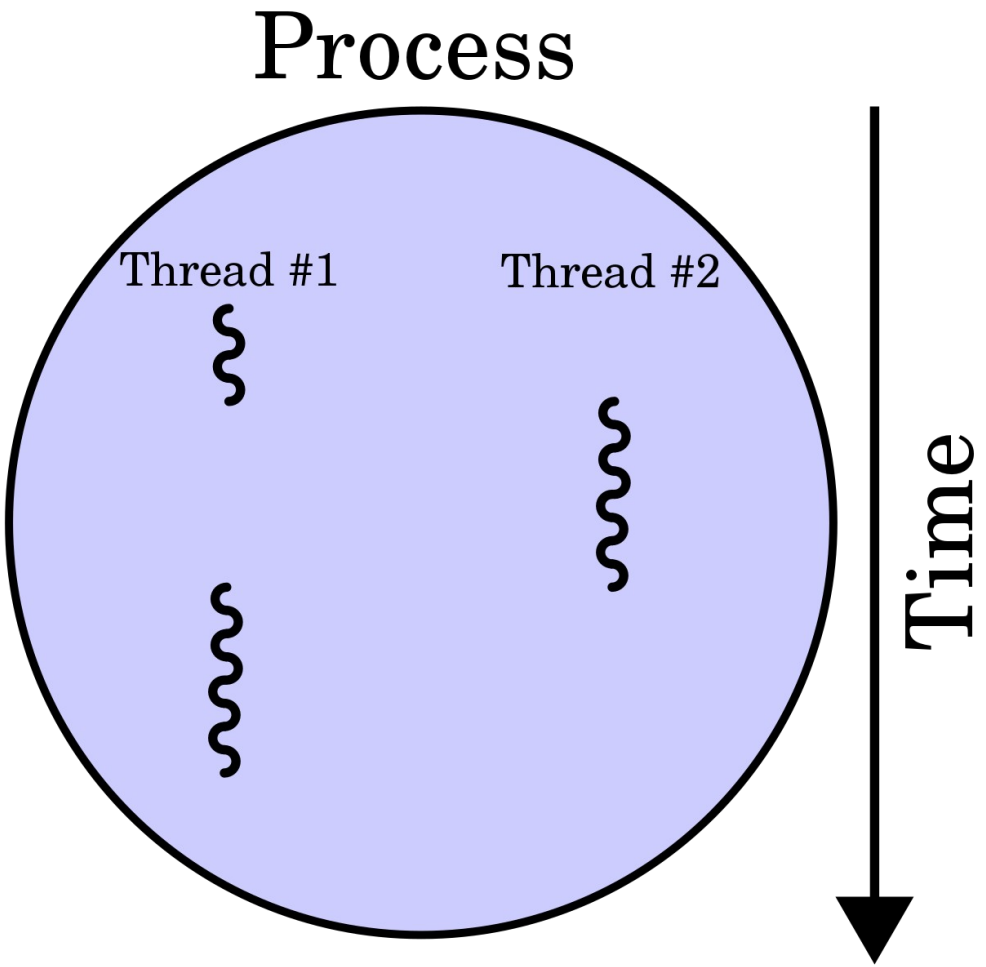
- Primjer:
  - Pokrenuto više istih Aplikacija – više procesa i više nezavisnih alokacija memorije
  - Prebacivanje iz jednog procesa u drugi zahtjeva vrijeme
  - Nezavisnost procesa je bitna odlika Operativnog sustava
    - Lakše je pronaći jedan proces koji je „zapeo” nego „gasiti cijelo računalo”

Osnovni pojmovi: Dretva

- Dretva (Thread)
  - Izvršavaju se unutar procesa (kod većine OS-a)
  - Dijele resurse (Memorija tj. adresni prostor)
    - Istoj varijabli mogu pristupiti različite dretve
  - Imaju zaseban stog (Stack)
  - Na sustavu s jednim procesorom događa se time-division multiplexing
    - Procesorsko vrijeme se prebacuje s jedne na drugu dretvu
    - Korisnik ima dojam da se operacije izvode simultano
  - Na višeprocesorskim sustavima dretve se izvode simultano ovisno o broju procesora / jezgara
- Svaki process ima adresni prostor i jednu kontrolnu dretvu

Proces vs. Dretva

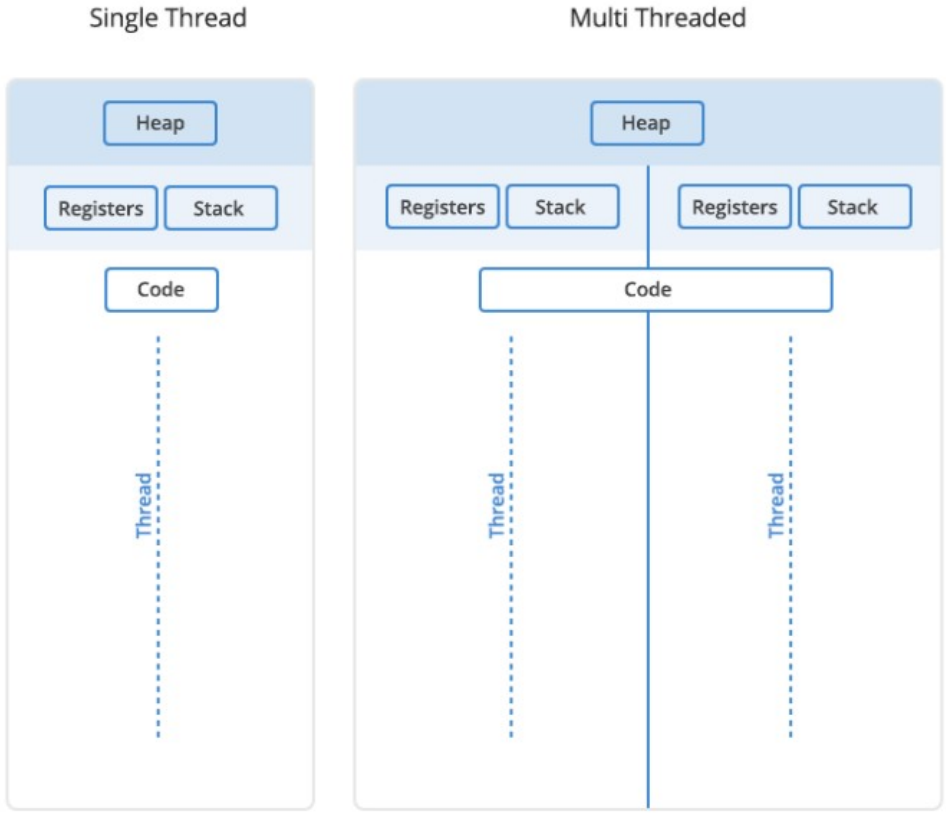
Svojstveno pojedinom procesu	Svojstveno pojedinoj dretvi
Adresni prostor	Programski brojač
Globalne varijable	Registri
Otvorene datoteke	Stog
Procesi djeca	Stanje
Tekući alarmi	
Signali i rukovatelji signalima	



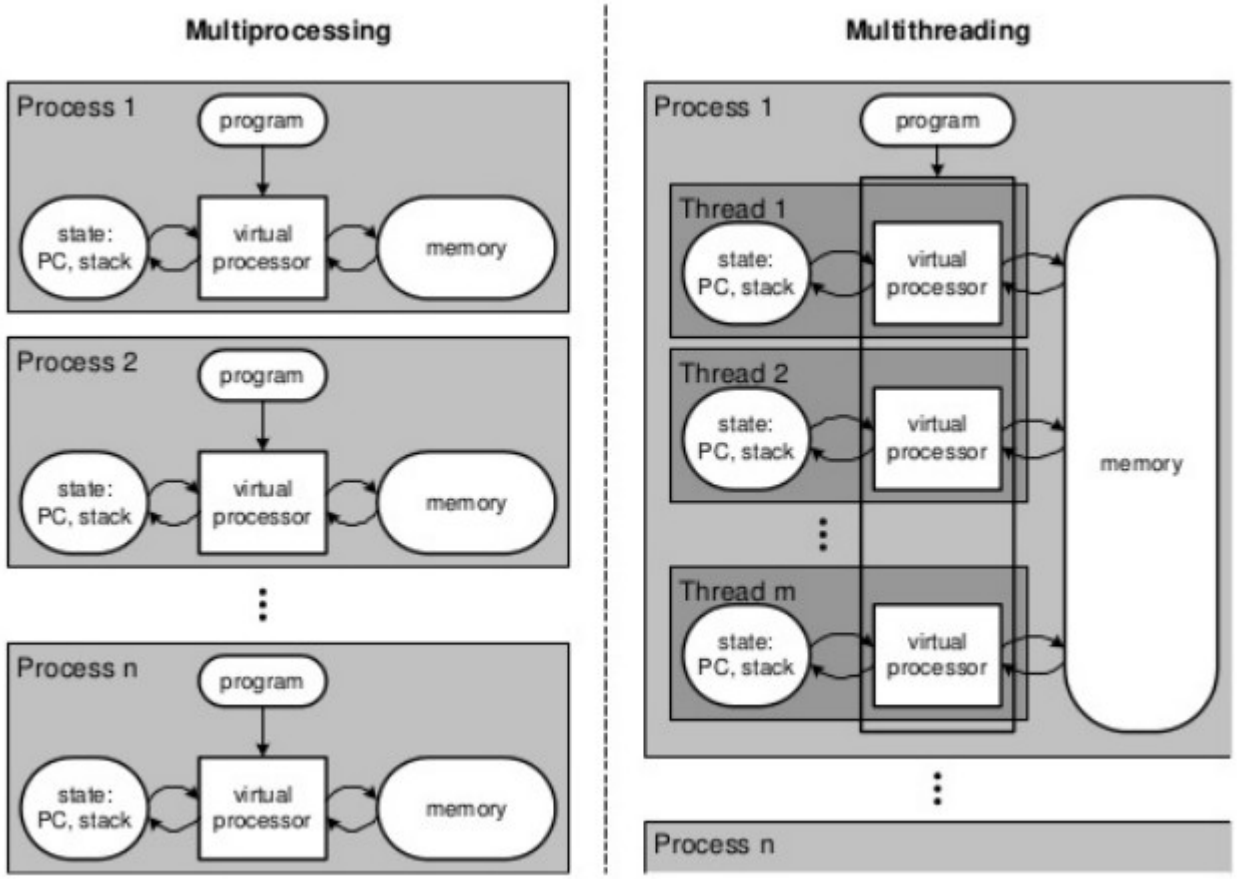
Višedretvenost primjer: jedna dretva čita sa diska, a druga vrši aritmetičke operacije

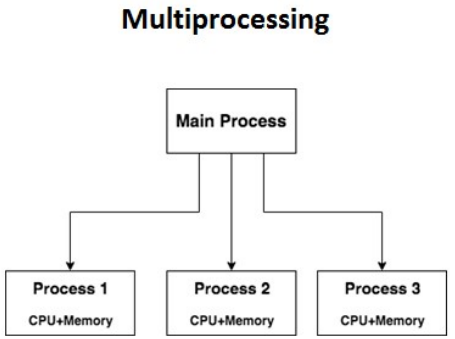
Dretvenost (Thread)

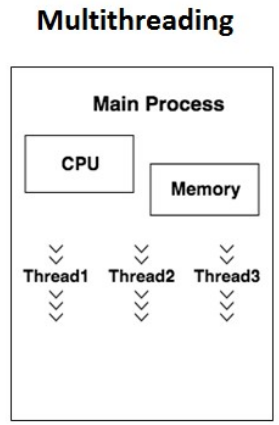
- Jedan proces može imati jednu ili više dretvi



Proces vs. Dretva







Nasljedna svojstva procesa

- Svaki proces može biti proces „roditelj” koji proizvodi svoju „djecu”
- Jezgra sustava, init process ima pid 0 i nema roditelja
- Svi ostali procesi imaju roditelje da li je to neki proces iz donjih dijelova hijerarhije ili jezgra operacijskog sustava
- Dijete dobiva kopije segmenta instrukcija i segmenta podataka roditelja
- Kada dijete završi svoj posao šalje svom roditelju: SIGCHILD signal da je gotov

Zombie proces

- Poznat još po imenu defunct proces
- Proces koji je završio izvršavanje (oslobodio je svoje resurse) ali se još uvijek nalazi u tablici procesa
- Zombie se pojavljuje kada „roditelj” ne sakupi svoje „dijete” po završetku izvršavanja (djete ne oslobodi PID)
  - Djete je završilo proces ali roditelj to nije priznao (djete je zombi)
- Ako roditelj prije završi tada dijete normalno nastavlja sa radom, a nadzor nad djetetom preuzima jezgra operacijskog sustava.
- *„Not dead and not alive”*

Orphan proces

- Proces koji se nastavlja izvoditi, a čiji je roditelj završio s radom
  - Tko će potvrditi da je proces djete završilo?
- Nadzor nad njim preuzima jezgra operacijskog sustava

Pokretanje novog procesa: Fork()

- Operacija kojom tekući proces (otac) kreira kopiju sebe (dijete)
- Vrijednosti:
  - -1 = NEISPRAVNO
  - 0 = Dijete proces uspješno kreiran
  - Ostalo > 0 = Tata od djeteta
- Kada su oba procesa kreirana nastavljaju izvršavanje daljnjih naredbi

Primjer 1

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{

    // make two process which run same
    // program after this instruction
    fork();

    printf("Hello world!\n");
    return 0;
}
```

Ispis:

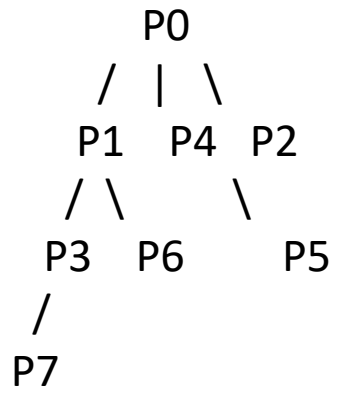
```
Hello World!
Hello World!
```

Primjer 2

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    fork();
    fork();
    fork();
    printf(„Hello World\n");
    return 0;
}
```

Ispis:

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```



P0 – glavni proces
1 fork: P1
2 fork P2,P3
3 fork: P4, P5, P6, P7

Primier 3

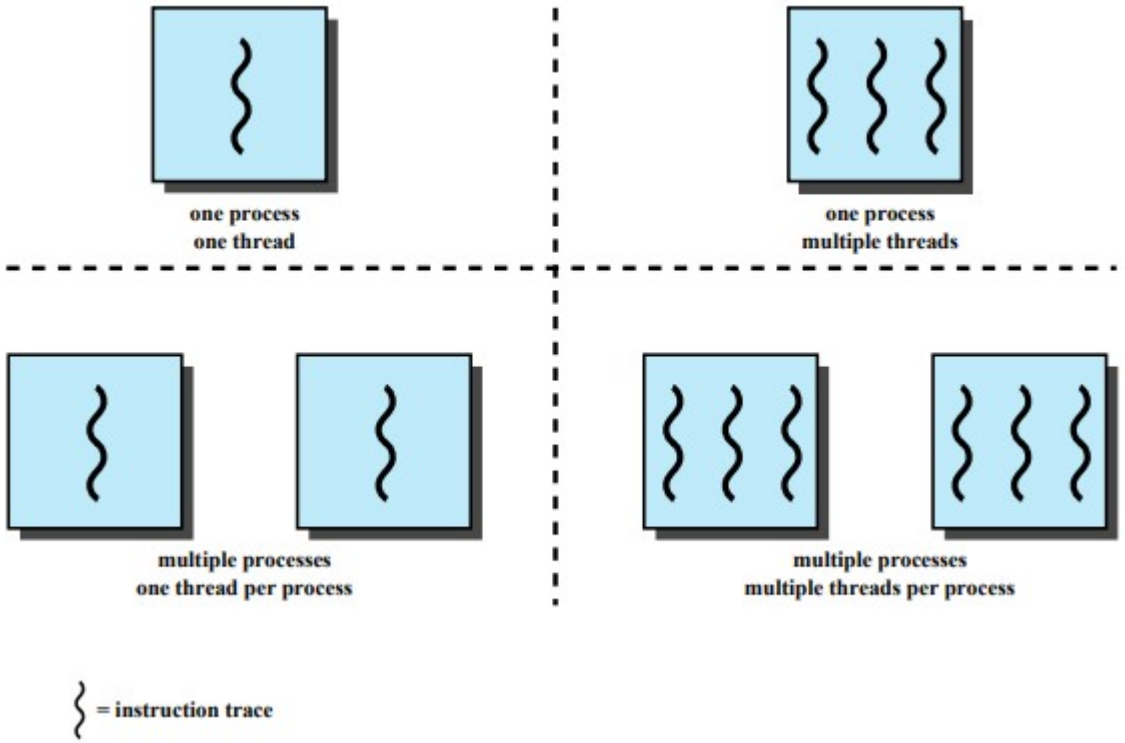
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

void  main(void)
{
    pid_t  pid,rez;
    int    i;

for (i=1;i<=2;i++){
   rez=fork();
   pid = getpid();
   if(rez==-1){
      printf ("Rez=%d A- Ovo je proces (PID) %d kojeg nisam uspio kreirati \n",rez, pid);
   } else if (rez==0){
      printf("Rez=%d B- Ovo je proces Djete/CHILD (PID)=%d, i=%d \n",rez, pid, i);
   } else{
      printf("Rez=%d C- Ovo je proces TATA/Parent (PID)=%d, i=%d \n",rez, pid, i);
      //exit(0);
          }
  }
 //rez=fork();
 //rez=fork();
}
```

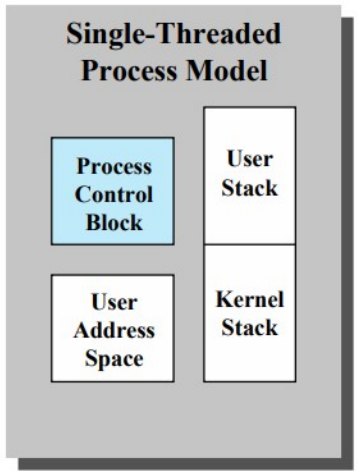
Ispis:
(ne mora nužno biti isti!)

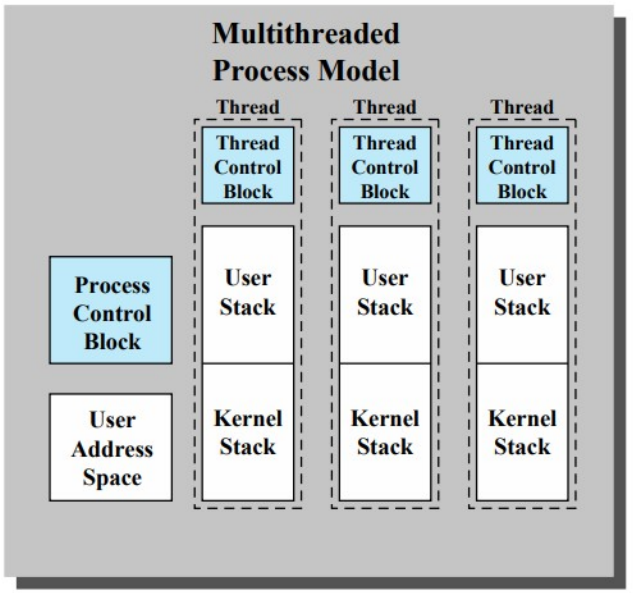
```
Rez=2655 C- Ovo je proces TATA/Parent (PID)=2654, i=1
Rez=2656 C- Ovo je proces TATA/Parent (PID)=2654, i=2
Rez=0 B- Ovo je proces Djete/CHILD (PID)=2655, i=1
Rez=2657 C- Ovo je proces TATA/Parent (PID)=2655, i=2
Rez=0 B- Ovo je proces Djete/CHILD (PID)=2655, i=1
Rez=0 B- Ovo je proces Djete/CHILD (PID)=2657, i=2
Rez=2655 C- Ovo je proces TATA/Parent (PID)=2654, i=1
Rez=0 B- Ovo je proces Djete/CHILD (PID)=2656, i=2
```

I3. CPU vrijeme i koncept dretvi

Dretveni pristupi







Dretve

- Koje su ključne prednosti korištenja dretvi? (proučiti predavanje!)
- U OSu koji podržava dretve – dodjeljivanje procesorskog vremena se obavlja na bazi dretve

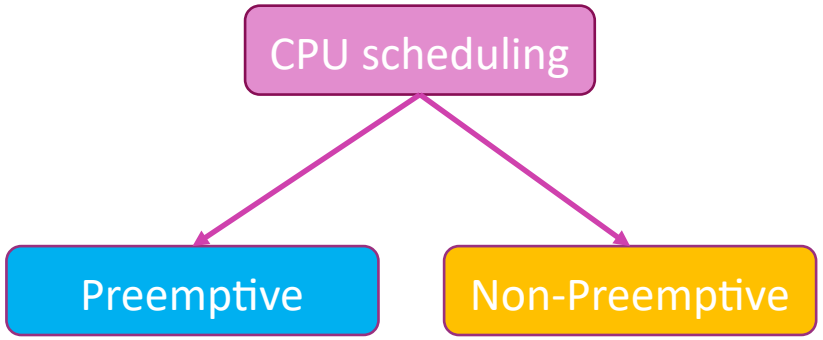
Ključna stanja dretve:
- Running
- Ready
- Blocked

Operacije dretvi povezane s promjenom stanja su:
- Spawn
- Block
- Unblock
- Finish

Detaljnije na predavanju!

Raspored izvršavanja dretvi u CPU-u

- Vrste višezadačnosti:
  - **Preemptive/Iznuđena**
    - Svaka dretva koja se izvodi **može** biti prekinuta u izvršavanju
    - **(-)** Manje efikasna od neiznuđene (više zamjena konteksta)
    - Svi moderni operacijski sustavi koriste ovu vrstu višezadaćnosti
  - **Non-Preemptive/Neiznuđena**
    - Kada neka dretva započne sa izvođenjem može se izvoditi sve dok ne završi osim ako ona sama ne dodijeli CPU drugoj dretvi prelaskom u stanje čekanja ili samostalnim iniciranjem pauze (*sleep*)
    - **(+)** Efikasnija i lakša za implementaciju
    - **(-)** Otvara problem beskonačno duge operacije (smrzavanja)

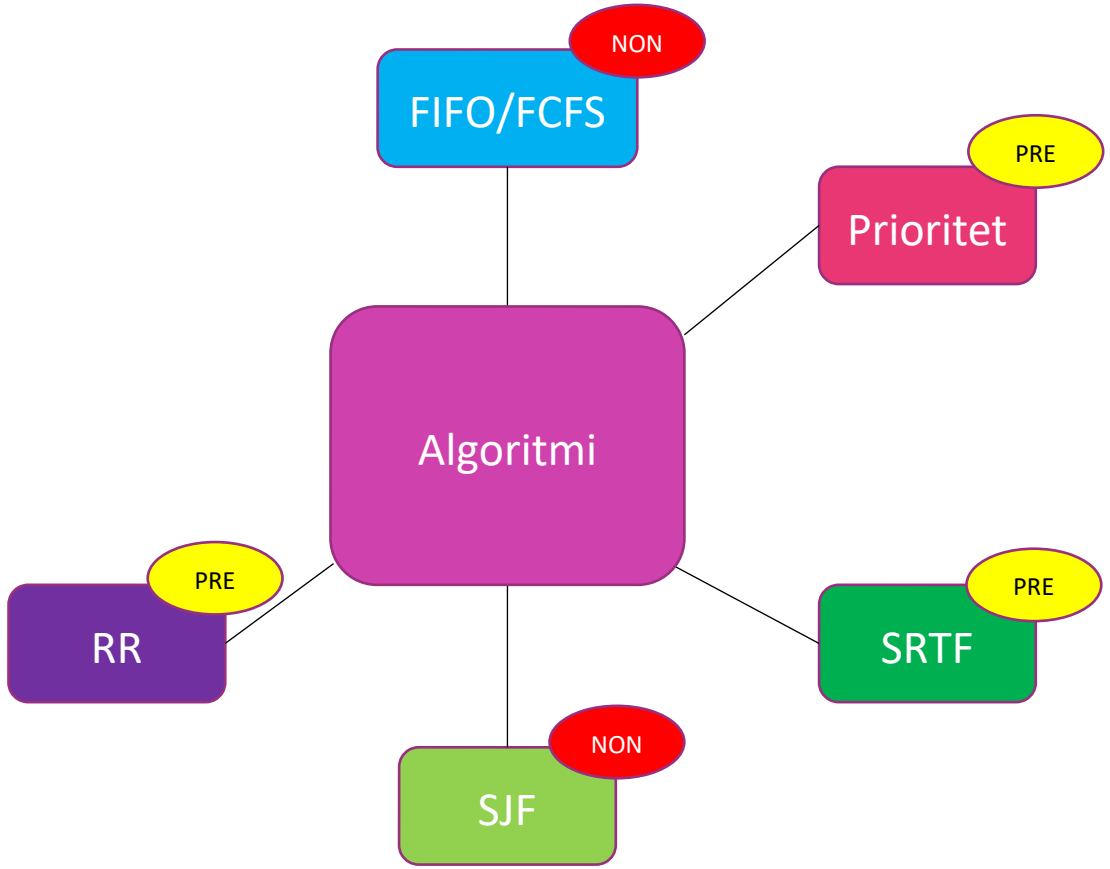


Ciljevi:

- Minimiziranje vremena odaziva
- Maksimiranje broja korisnih operacija u CPUu
- Pravednost

Algoritmi za raspodjelu CPU vremena

- Algoritmi:
  - Frist In Frist Out (FIFO) or FCFS First Come First Serve (FCFS)
  - Prioritet
  - Shortest Remaining Time Frist (SRTF)
  - Shortest-Job-First (SJF) Scheduling
  - Round Robin Scheduling (RR)
  - Multilevel Queue Scheduling (MLQS)




Homework

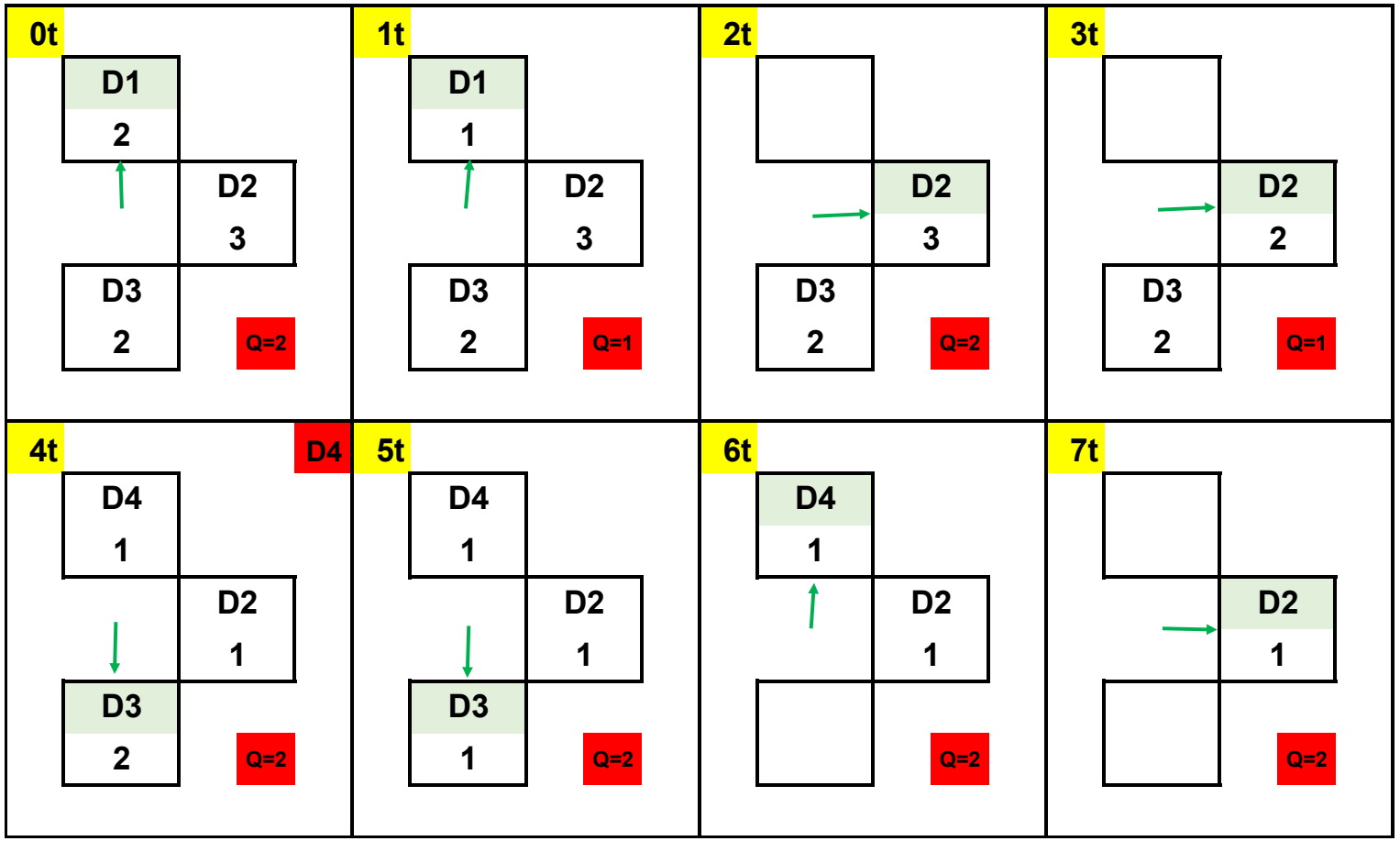
Zadaci za vježbu 1

- Prikažimo sva stanja algoritma do kraja svih dretvi, ako je trenutno stanje reda D1=2t, D2=3t, D3=2t, u 0t trenutku je CPU slobodan i kazaljka izvršavanja treba prijeći na D1 dretvu.
- Naknadno u 4t dolazi nova dretva u red D4=1t
- Algoritmi:
  a) FIFO/FCFS
  b) RR, Q=1
  c) RR, Q=2
  d) SJF
  e) SRTF, Interrupt/Prekid = 1t

Rješenje 1

a) FIFO - Prosječno vrijeme čekanja = 2,25t
b) RR (Q=1) - Prosječno vrijeme čekanja = 3,25t
c) RR (Q=2) - Prosječno vrijeme čekanja = 2,75t
d) SJF - Prosječno vrijeme čekanja = 1,75t
e) SRTF - Prosječno vrijeme čekanja = 1,75t

Zadatak 1c



8t-EMPTY

Average waiting time:				
Prosječno vrijeme čekanja:				
D1=	0			
D2=	5			
D3=	4			
D4=	2			
	11	2.75	t	

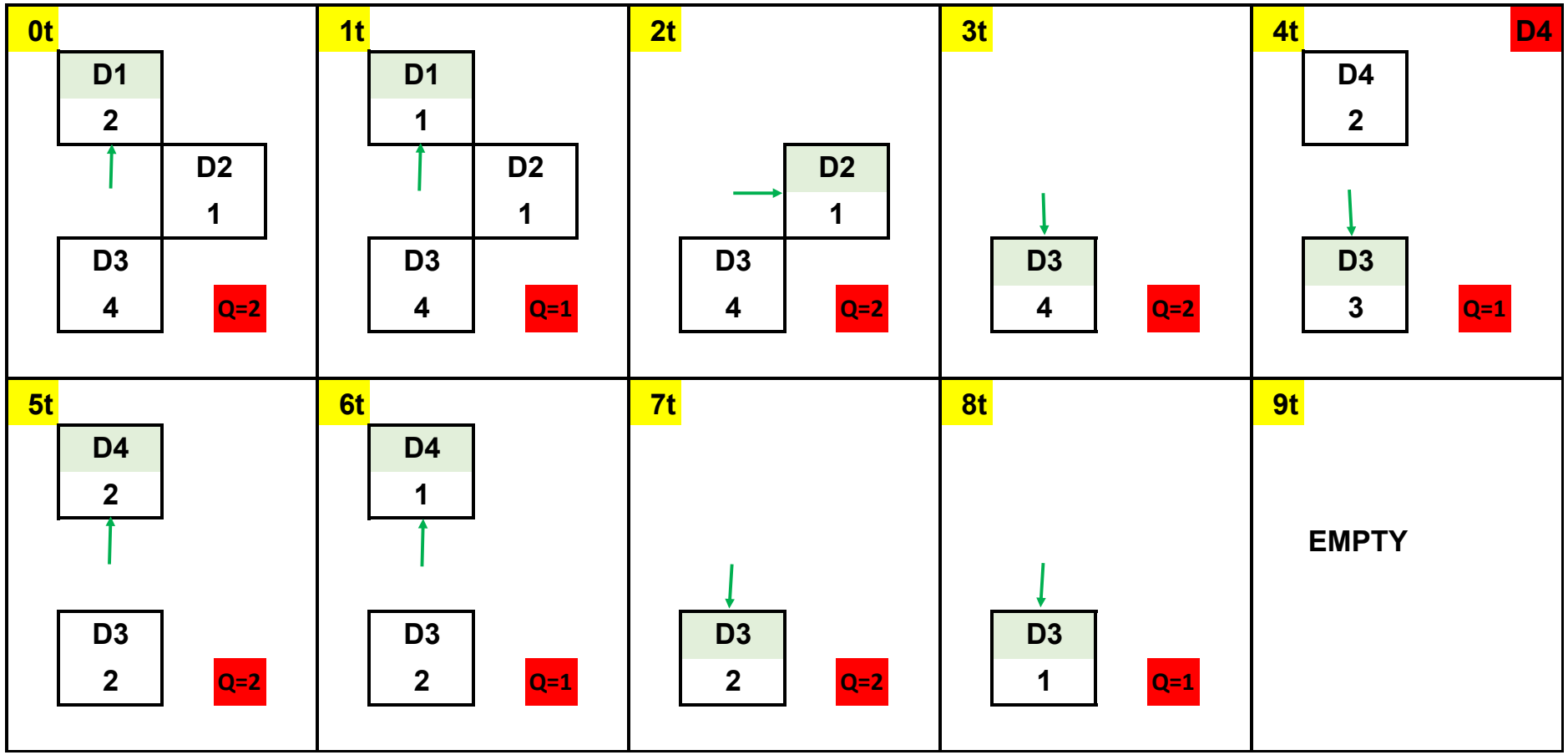
Zadaci za vježbu 2

- Prikažimo sva stanja algoritma do kraja svih dretvi, ako je trenutno stanje reda D1=2t, D2=1t, D3=4t, u 0t trenutku je CPU slobodan i kazaljka izvršavanja treba prijeći na D1 dretvu.
- Naknadno u 4t dolazi nova dretva u red D4=2t
- Algoritmi:
  a) FIFO/FCFS
  b) RR, Q=2
  c) SJF
  d) SRTF, Interrupt/Prekid = 1t
  e) SRTF, Interrupt/Prekid = 2t

Rješenje 2

a) FIFO - Prosječno vrijeme čekanja = 2t
b) RR - Prosječno vrijeme čekanja = 2t
c) SJF - Prosječno vrijeme čekanja = 1,75
d) SRTF (I=1) - Prosječno vrijeme čekanja = 1,5t
e) SRTF (I=2) - Prosječno vrijeme čekanja = 1,75t

Rješenje 2b



Average waiting time:

Prosječno vrijeme čekanja:

D1= 0

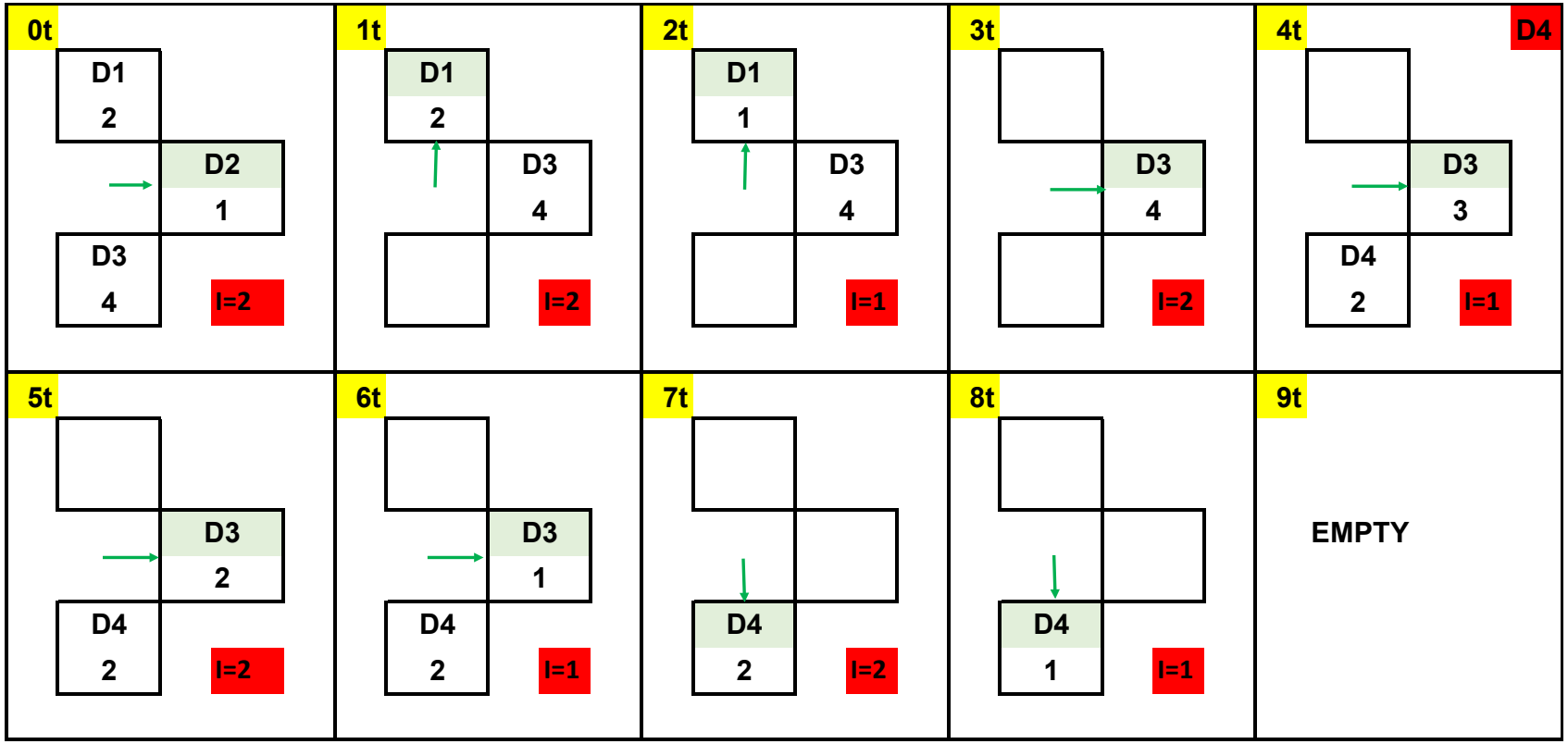
D2= 2

D3= 5

D4= 1

8 2 t

Rješenje 2e



Average waiting time:

Prosječno vrijeme čekanja:

D1= 1

D2= 0

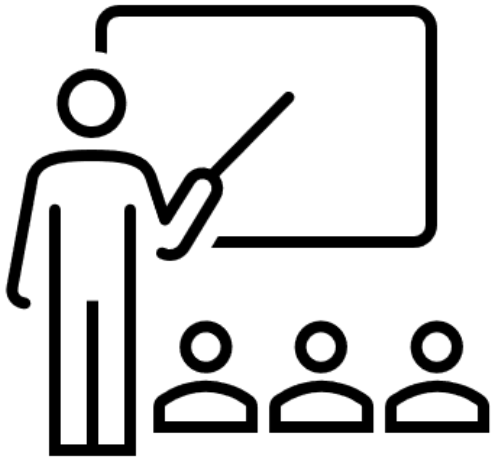
D3= 3

D4= 3

7 1.75 t

Ponovite još…

- Kernel OSa
- Boot proces
- Prsteni
- Multitasking vs. Multiprocessing
- Vježbajte zadatke
- …



- Prezentacije sa predavanja (ne samo sa vježbi)
- Pročitajte knjigu (ili druge izvore na Internetu)

Hvala na pažnji!

Sretno na među-ispitu!