Operativni sistemi 1

Konkurentno programiranje

Komunikacija i sinhronizacija

Predmetni asistenti:

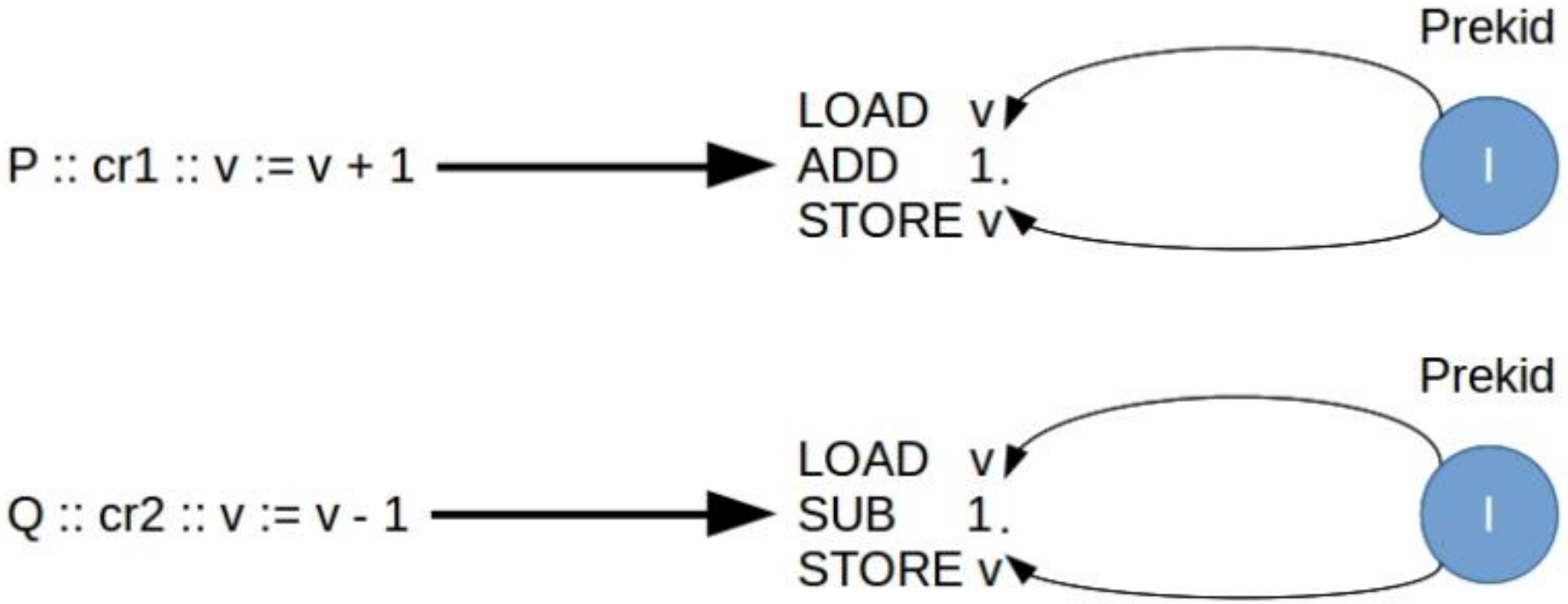
Mihailo Obrenović, Jelica Vasiljević

Pristup zajedničkom resursu

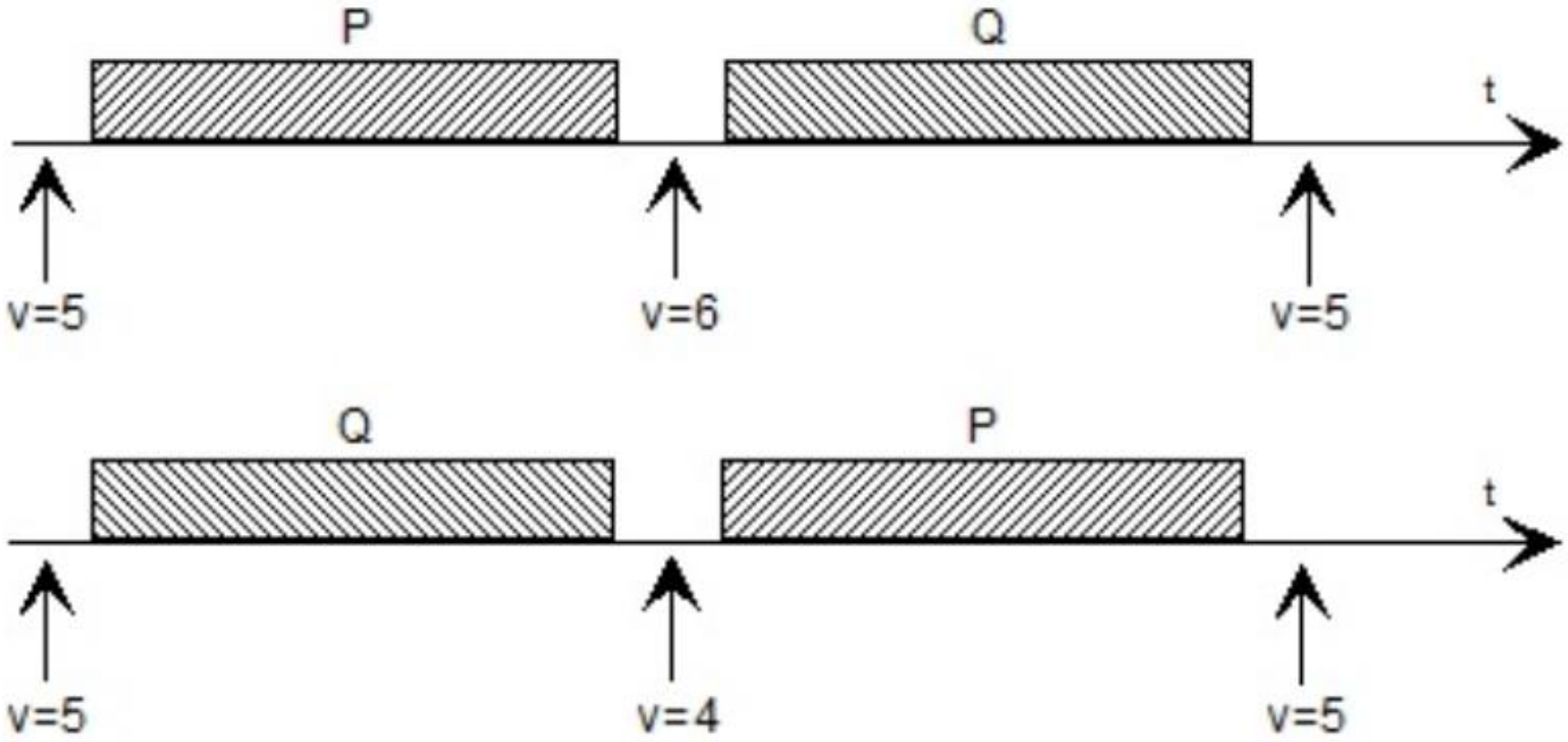
- P, Q - procesi
- Zajednički resurs – deljena promenljiva, bafer, ...
- Kritični region – deo koda koji pristupa zajedničkom resursu (cr1 i cr2 u P i Q)

```
P ::        Q ::
s11;        s21;
cr1;        cr2;
s12;        s22;
```

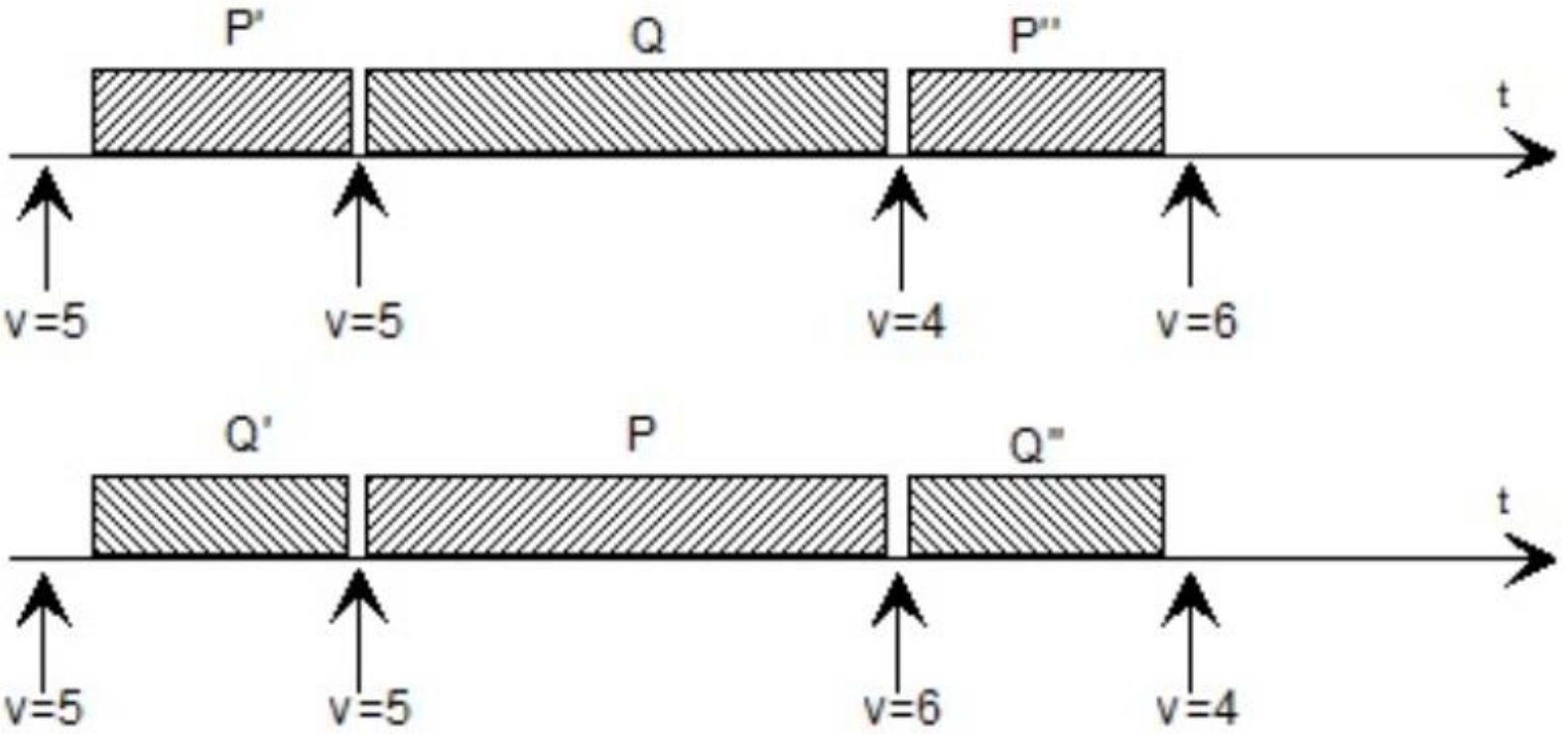
Problem deljene promenljive



Sekvencijalno izvršavanje

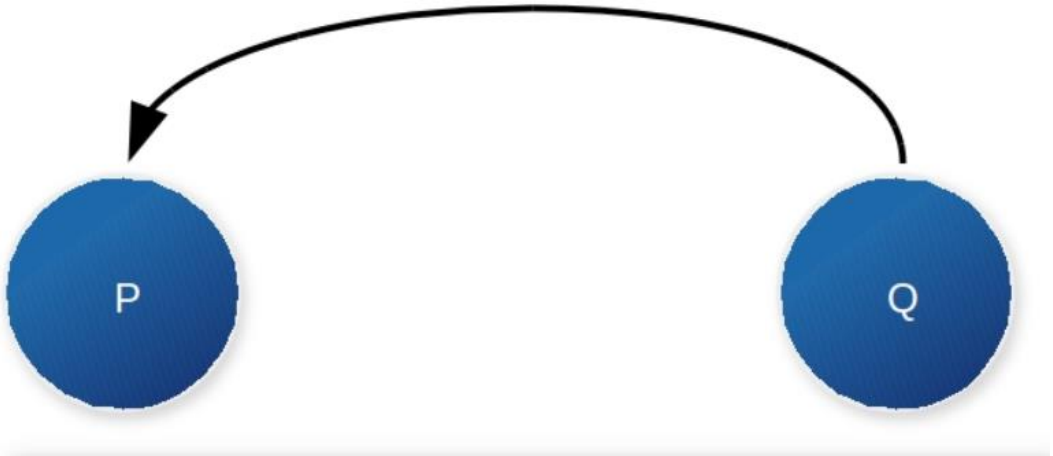


Izvršavanje sa prekidima u kritičnom regionu



- **Zaključak:** Ne sme se dozvoliti da dva procesa istovremeno pristupe zajedničkim promenljivama (da uđu istovremeno u kritični region).

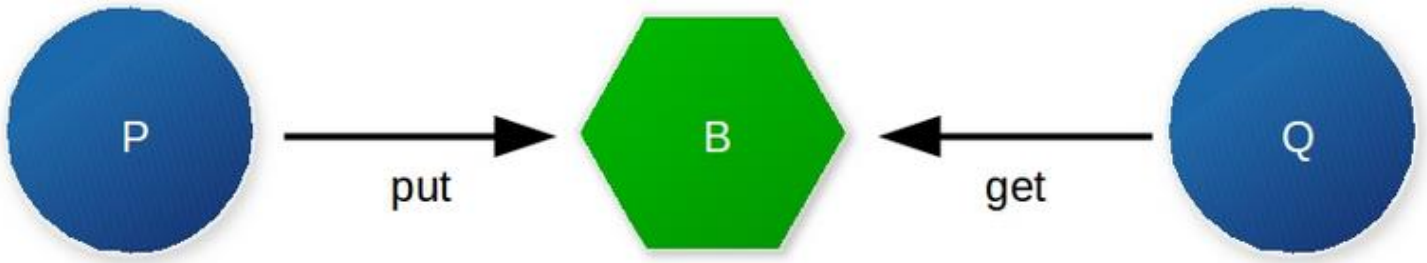
Problem signalizacije



- Ponekad je potrebno da proces P sačeka sa izvršenjem nekog segmenta koda dok proces Q ne završi određeni posao

```
P ::                          Q ::
s11;                          s21;
Čekaj na signal iz Q;         Signaliziraj P da može da nastavi;
s12;                          s22;
```

Problem primopredaje podataka



- P – proces Producer
- Q – proces Consumer
- B – bafer za razmenu podataka
- Sihronizacija proizvodnje i utroška

P ::

```
var x : slot;
while true do
begin
  proizvedi x;
  put(x);
end;
```

Q ::

```
var y: slot;
while true do
begin
  get(y);
  utrosi y;
end;
```

Jednostruki bafer

```
type slot = array 1 .. M of integer;
```

```
var B : slot;
```

prostor bafera

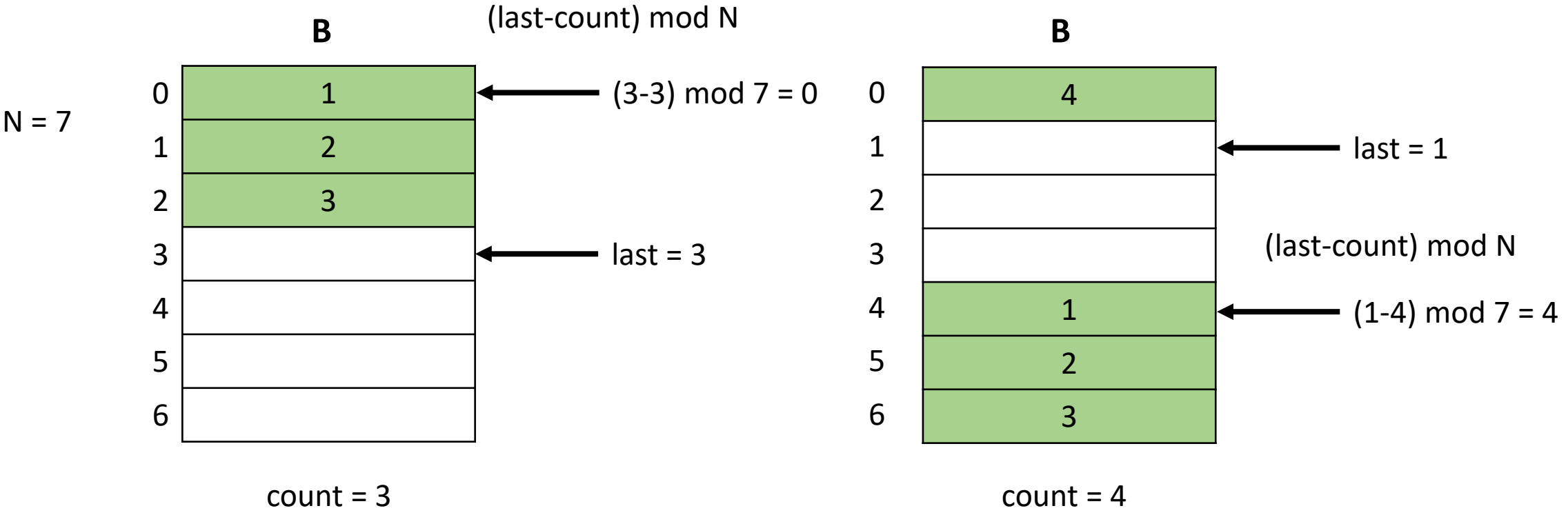
```
procedure put (x : slot)
var i : integer;
begin
  for i := 1 to M do B[i] := x[i];
end;
```

Operacija stavljanja podataka u bafer.

```
procedure get (var x : slot)
var i : integer;
begin
  for i := 1 to M do x[i] := B[i];
end;
```

Operacija uzimanja podataka iz bafera.

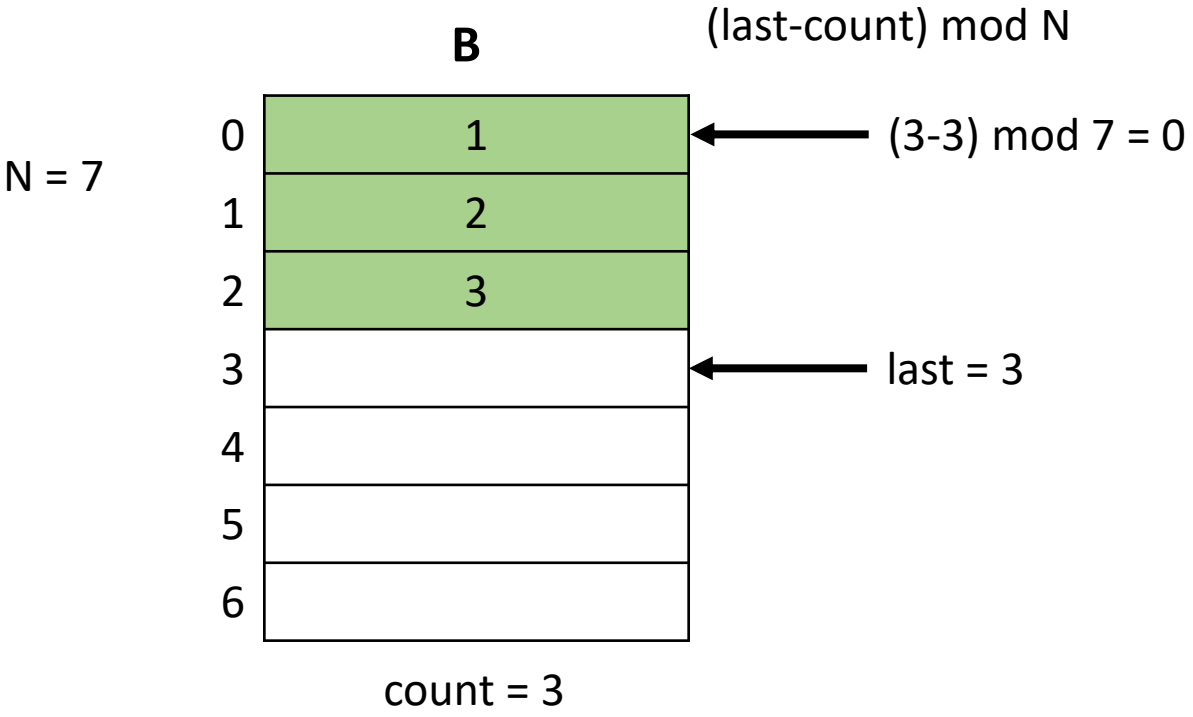
Višestruki (konačni) bafer

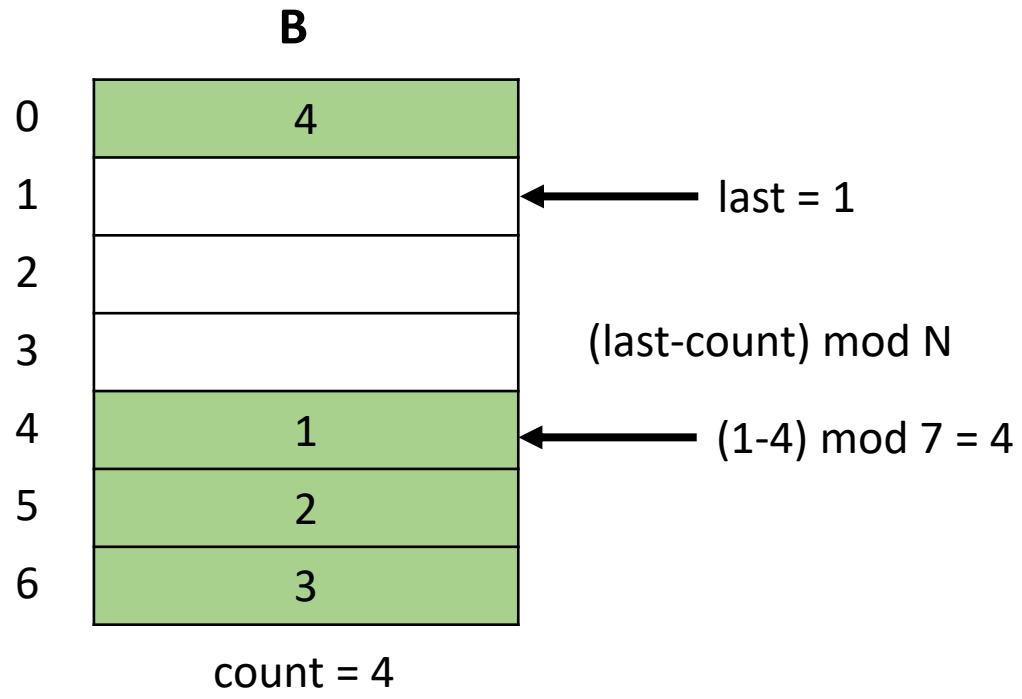


- Proces P proizvodi podatke za proces Q
- Bafer se popunjava kružno

- Last – kraj bafera
- Count – brojač bafera
- (last-count) mod N – određivanje početka bafera

Višestruki (konačni) bafer





- P ne može da šalje podatke u pun bafer
- Q ne može da čita podatke iz praznog bafera
- P čeka da Q isprazni bafer
- Q čeka da P napuni bafer

- Q treba da obavesti P da je ispraznio bafer
- P treba da obavesti Q da je napunio bafer
- **Kritični region + sinhronizacija**

Višestruki (konačni) bafer

```
procedure put (x : slot)
var i : integer;
begin
  for i := 1 to M do B[last, i] := x[i];
  last := (last + 1) mod N;
  count := count + 1;
end;
```

Operacija stavljanja podataka u bafer.

```
procedure get (var x : slot)
var i : integer;
begin
  for i := 1 to M do x[i] := B[(last - count) mod N, i];
  count := count - 1;
end;
```

Operacija uzimanja podataka iz bafera.

Načini zaštite kritičnog regiona

1) Zabrana prekida

P

```
s11;
disable;
cr1;
enable;
s12;
```

zabrana prekida
kritični region
ukidanje zabrane prekida

Q

```
s21;
disable;
cr2;
enable;
s22;
```

Aplikacioni programer mora
da pazi da kritične regione
obavezno "opkoli" parom
mašinskih naredbi DISABLE/ENABLE.

- **Problemi**

- Dugačak kritičan region, prekidi zabanjeni na duže vreme, onemogućeno multiprocesiranje, traćenje procesorskog vremena
- Spuštanje na asemblerski nivo programiranja
- Metod nije dovoljan u slučaju multiprocesorskih sistema – zabrana prekida je na nivou jednog procesora

Načini zaštite kritičnog regiona

2) Čekanje u petlji (Busy Wait)

P ::

```
s11;
while l do w1;
l := true;
cr1;
l := false;
s12;
```

Q ::

```
s21;
while l do w2;
l := true;
cr2;
l := false;
s22;
```

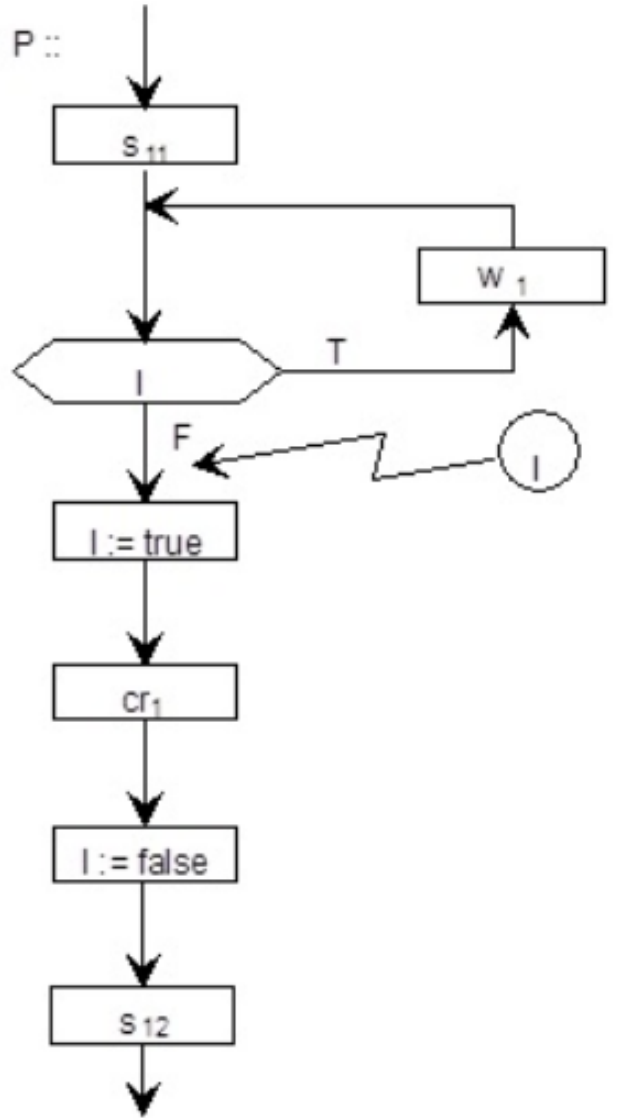
- `l` – logički indikator
  - `l=true` – kritični region zauzet
  - `l=false` – kritični region slobodan
- `w1`, `w2` – *sleep* naredbe

- Pre ulaska u kritični region se proverava da li je kritični region zauzet
- Ako je zauzet, proces ulazi u petlju u kojoj izvršava *sleep* naredbe određene dužine sve dok se kritični region ne oslobodi (`while l do w1`)
- Kada je kritični region slobodan, proces ga označava zauzetim (`l:=true`) izvršava naredbe kritičnog regiona (`cr1` ili `cr2`), a onda ga označava slobodnim (`l:=false`)

Načini zaštite kritičnog regiona

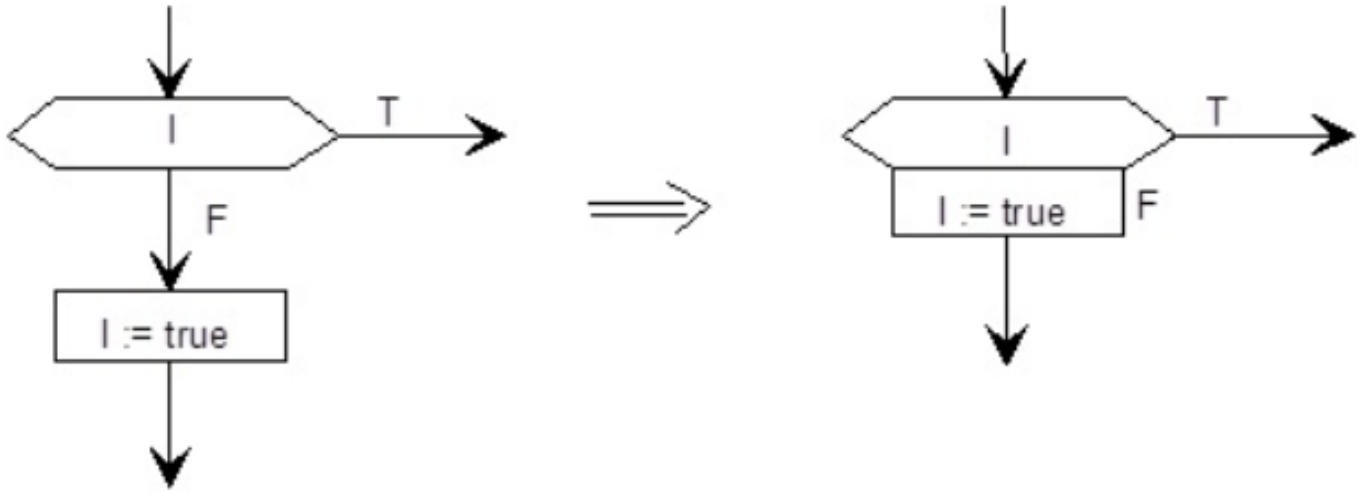
- **Problem**

- Proces P primi prekid između provere da li je kritični region zauzet i zauzimanja kritičnog regiona
- Za vreme prekida, proces Q može zauzeti kritični region
- Dok je Q još uvek u kritičnom regionu, proces P može da nastavi sa radom i da takođe uđe u kritični region
- Dva procesa istovremeno u kritičnom regionu!



Načini zaštite kritičnog regiona

- **Rešenje – TEST-AND-SET naredba**




```
function testset(var l : boolean)
: boolean;
begin
  testset := l;
  l := true;
end;
```

Ova funkcija predstavlja jednu mašinsku naredbu (koja ne može da se prekine)

P ::

```
s11;
while testset(l) do w1;
cr1;
l := false;
s12;
```

Q ::

```
s21;
while testset(l) do w2;
cr2;
l := false;
s22;
```

Problemi

- Traćenje procesorskog vremena u beskonačnoj petlji
- Spuštanje na asemblerski nivo programiranja (`testset`)
- Promenljiva `l` je deljena, a van kritičnog regiona – nije obezbeđena