1.Napisati URM program koji izračunava funkciju $f(x,y) = x + y$

Zbir posmatramo kao uzastopnu primenu operacije sledbenik, tj. operacije dodavanja jedinice

Početna konfiguracija

```
------------------------
| x | y |
------------------------
```

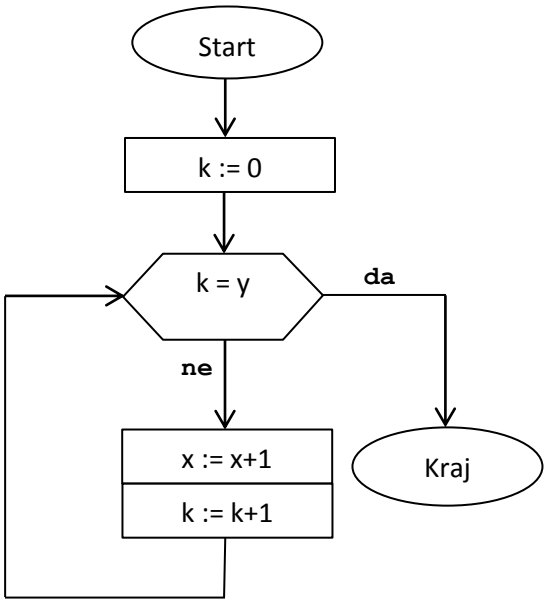
Konfiguracija u toku rada – k broji koliko smo do tog trenutka dodali jedinica, kada dostigne vrednost y, napravili smo traženi zbir

```
------------------------
| x + k | y | k |
------------------------
```

Završna konfiguracija

```
------------------------
| x + y | y | y |
------------------------
```

Dijagram toka



URM program:

1. **Z(3)**
2. **J(3,2,100)**
3. **S(1)**
4. **S(3)**
5. **J(1,1,1)**

2. Napisati URM program koji izračunava funkciju $f(x,y) = \begin{cases} 0, ako\ je\ x <= y \\ 1, ako\ je\ x > y \end{cases}$

Početna konfiguracija

```
------------------------
| x | y |
------------------------
```

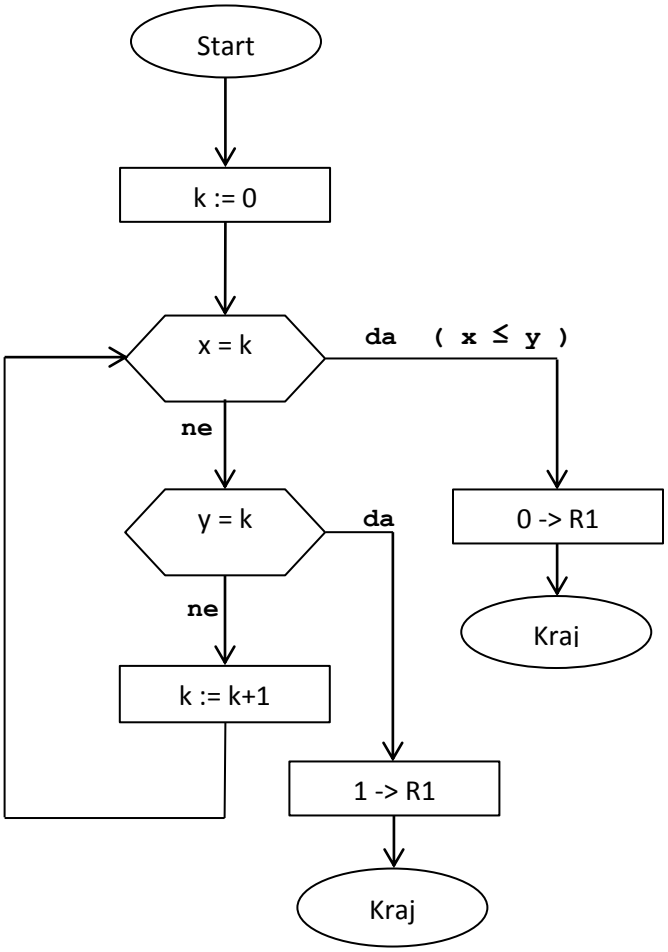
Konfiguracija u toku rada – k je brojač koji koristimo za poredjenje sa x, odnosno sa y, prva dostignuta vrednost je broj koji je manji ili jednak drugom

```
--------------------------------
| x | y | k |
--------------------------------
```

Završna konfiguracija

```
-------------------------------------------------
| 0 ili 1 u zavisnosti od rezultata poredjenja|
-------------------------------------------------
```

Dijagram toka



URM program:

```
1. Z(3)
2. J(1,3,6)
3. J(2,3,8)
4. S(3)
5. J(1,1,1)
6. Z(1)
7. J(1,1,100)
8. Z(1)
9. S(1)
```

3. Napisati URM program koji izračunava funkciju $f(x,y) = \begin{cases} x-y, & ako\ je\ x > y \\ 0, & ako\ je\ x <= y \end{cases}$

Početna konfiguracija

x	y

Konfiguracija u toku rada – k je brojač koji koristimo za poređenje i računanje razlike.

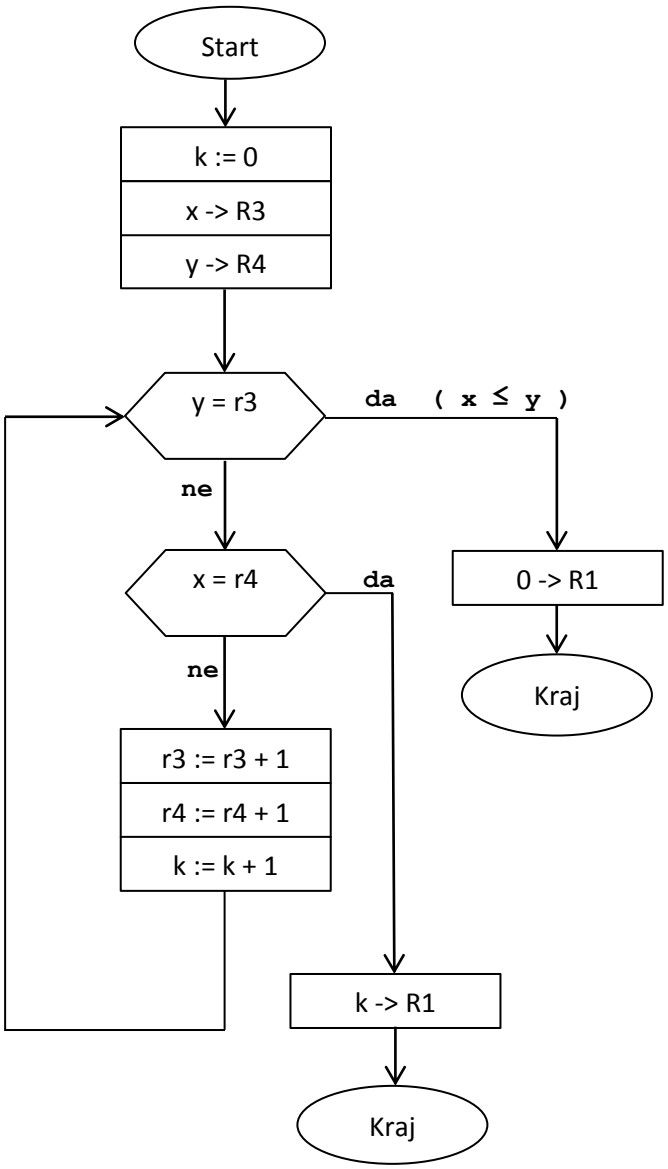
x	y	x + k	y + k	k

U svakom koraku simultano poredimo broj x + k sa y i broj y + k sa x, koja vrednost se prva dostigne (y ili x) je broj ne manji od drugog – ukoliko je to x, brojač k je ujedno tražena razlika, pa tu vrednost prebacujemo u prvi registar kao rezultat izračunavnja, a ukoliko je to y, treba kao rezultat postaviti nulu u prvi registar.

URM program:

```
1. Z(5)
2. T(1,3)
3. T(2,4)
4. J(2,3,10)
5. J(1,4,12)
6. S(3)
7. S(4)
8. S(5)
9. J(1,1,4)
10. Z(1)
11. J(1,1,100)
12. T(5,1)
```

Dijagram toka



4. Napisati URM program koji izračunava funkciju $f(x, y) = x * y$

Proizvod svodimo na zbir

x*y = x + x + ... + x

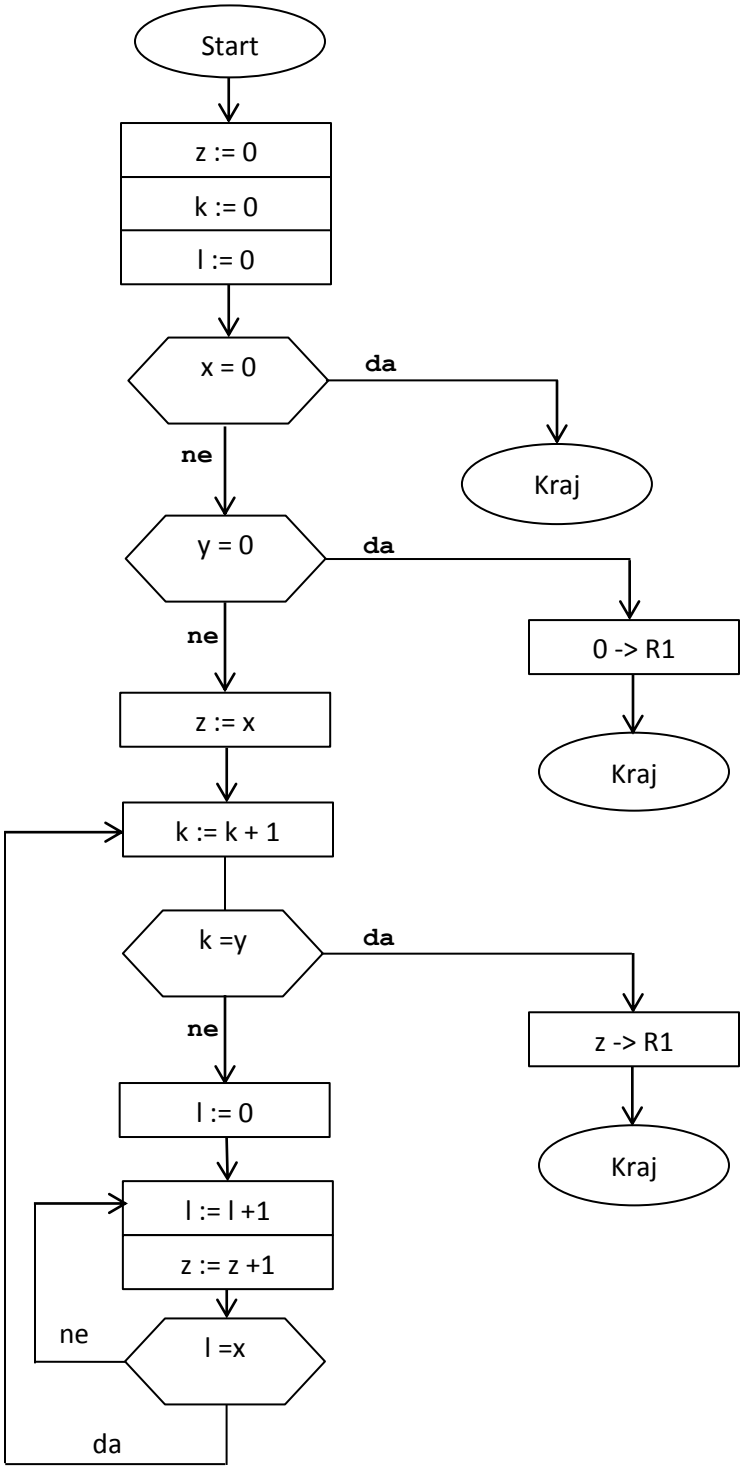
k nam služi da znamo koliko smo puta do tog trenutka dodali broj x i kada k dostigne vrednost y znamo da je napravljen traženi proizvod.

Zbir svodimo na uzastopnu primenu operacije sledbenik – podsetiti se korišćenog algoritma za funkciju zbir, zbog toga nam je potreban još jedan

brojač l koji nam služi da znamo koliko smo jedinica dodali (pri pravljenju broja x). Brojač l mora da se postavi na nulu pre svakog sledećeg sabiranja međurezultata sa novim x!

x*y = x + (1 + 1 + ... + 1) + ... + (1 + 1 + ... + 1)

Dijagram toka



Početna konfiguracija

```
------------------------
| x | y |
------------------------
```

Konfiguracija u toku rada – k je brojač za y, l brojač za x, u z pravimo proizvod – na kraju se mora prebaciti rezultat iz trećeg registra u prvi

```
------------------------
| x | y | z | k | l |
------------------------
```

URM program

```
1. Z(3)
2. Z(4)
3. Z(5)
4. J(1,3,100)
5. J(2,3,16)
6. T(1,3)
7. S(4)
8. J(4,2,14)
9. Z(5)
10.S(5)
11.S(3)
12.J(5,1,7)
13.J(1,1,10)
14.T(3, 1)
15.J(1,1,100)
16.Z(1)
```

5. Napisati URM program koji izračunava funkciju $f(x) = x^2 + 1$

Primetimo da je $x^2 = x * x$, tj. možemo iskoristiti algoritam za množenje – sada su x i y jednaki, odnosno oba brojača koja smo koristili u prethodnom zadatku uzimaju vrednosti od 0 do x, na kraju samo treba dobijeni proizvod uvećati za 1 pomoću instrukcije sledbenik

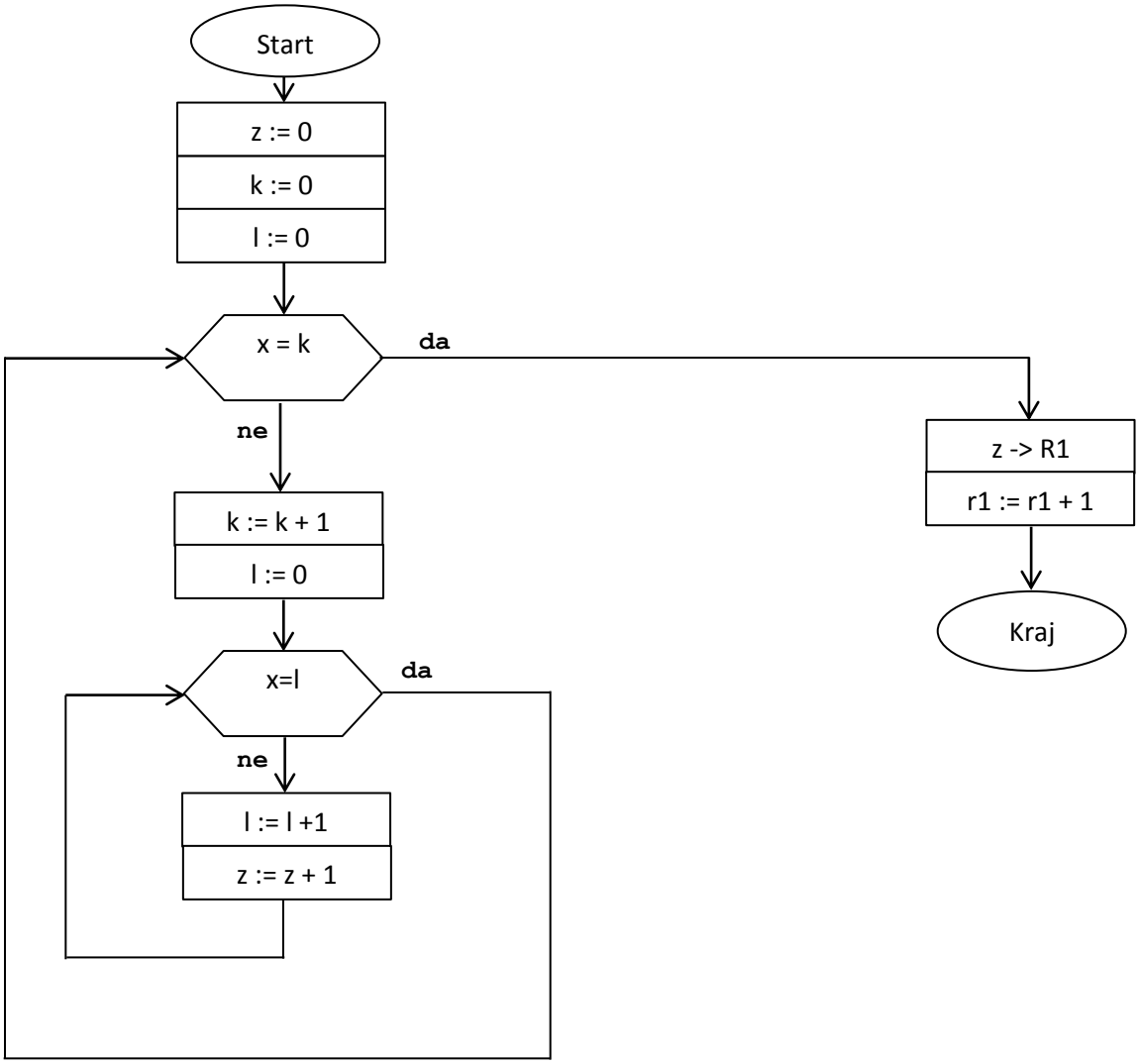
Početna konfiguracija

```
------------------------
| x |
------------------------
```

Konfiguracija u toku rada – k i l su brojači, u z pravimo proizvod, tj. x^2 – na kraju se mora prebaciti rezultat iz drugog registra u prvi i uvećati za 1

```
----------------------
| x | z | k | l |
----------------------
```

Dijagram toka



URM program

```
1. Z(2)
2. Z(3)
3. Z(4)
4. J(1,3,11)
5. S(3)
6. Z(4)
7. J(1,4,4)
8. S(4)
9. S(2)
10.J(1,1,7)
11.T(2,1)
12.S(1)
```

6. Napisati URM program koji izračunava funkciju $f(x) = x^2 + x + 1$

Prvi način: za vežbu napišite program koristeći prethodni zadatak – potrebno je dodati još deo za zbir $x^2 + 1$ i x

Drugi način: primetimo da je $x^2 + x + 1 = x * (x + 1) + 1$, sveli smo zadatak na računanje proizvoda dva broja – ovde je y = x + 1

Bitno: funkcija ima samo jedan argument, koji je podrazumevano u prvom registru. Sami pravimo $x + 1$ u drugom registru koji posle koristimo kao y iz programa za proizvod.

Početna konfiguracija

```
------------------------
| x |
------------------------
```

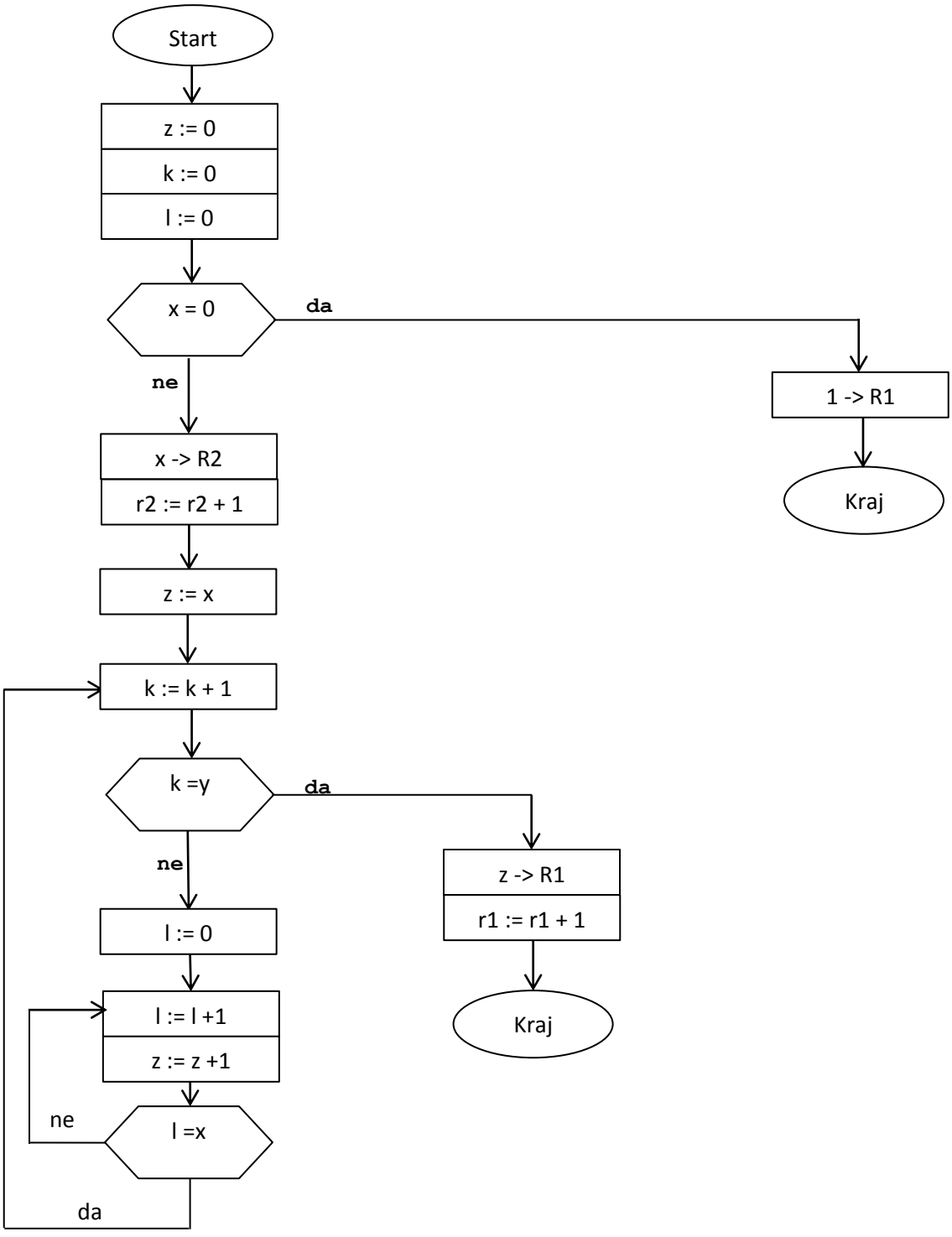
Konfiguracija u toku rada – k i l su brojači, u z pravimo proizvod $x * (x + 1)$ na kraju se mora prebaciti rezultat iz trećeg registra u prvi i uvećati za 1

```
------------------------
| x | x+1 | z | k | l |
------------------------
```

URM program

```
1. Z(3)
2. Z(4)
3. Z(5)
4. J(1,3,15)
5. T(1,2)
6. S(2)
7. T(1,3)
8. S(4)
9. J(4,2,18)
10.Z(5)
11.S(5)
12.S(3)
13.J(5,1,8)
14.J(1,1,11)
15.Z(1)
16.S(1)
17.J(1,1,100)
18.T(3,1)
19.S(1)
```

Dijagram toka



7.Za vežbu: Napisati URM program koji izračunava funkciju $f(x) = 3 * x + y$

Kada se u funkciji javlja konstanta, nju pravimo u registru koristeći nula instrukciju i instrukciju sledbenik.

Primer: URM program za pravljenje broja 3 u n-tom registru (n je redni broj registra gde želimo da smestimo konstantu)

1. Z(n)
2. S(n)
3. S(n)
4. S(n)

Zadatak se svodi na primenu već poznatih programa za proizvod i zbir, jedina razlika je što za proizvod sada imamo dat samo jedan argument dok drugi – broj 3 pravimo sami u slobodnom registru na prethodno opisan način.

8.Formiranje velikih brojeva:

Napisati URM program koji izračunava funkciju $f(x) = x + 121$

Kako napraviti neki veliki broj u programu, kao u ovom slučaju 121? Jedan način za pravljenje broja 121 jeste da se napiše nula instrukcija i 121 put instrukcija sledbenik – kako ovo oduzima puno vremena, bolja ideja je da se veliki broj preko nekih operacija svede na manje brojeve za koje nam nije teško napisati program.

Napomena: ne sme se napisati sledeće! Tačkice nisu deo URM programa! Ovakvo rešenje se ne priznaje!

```
1. Z(2)
2. S(2)
...
122. S(2)
```

Primetimo da je 121 = 11 * 11. Iskoristićemo program za proizvod, a broj 11 pravimo kao u primeru za broj 3.

Početna konfiguracija

```
------------------------
| x |
------------------------
```

Konfiguracija u toku rada – k i l su brojači za broj 11, u z prvo prebacimo x pa dalje pravimo proizvod 11*11. Na kraju se mora prebaciti rezultat iz trećeg registra u prvi.

```
------------------------
| x | 11 | z | k | l |
------------------------
```

URM program:

1. Z(2)
2. Z(3)
3. Z(4)
4. Z(5)
5. S(2)
6. S(2)
7. S(2)
8. S(2)
9. S(2)
10. S(2)
11. S(2)
12. S(2)
13. S(2)
14. S(2)
15. S(2)
16. T(1,3)
17. J(4,2,24)
18. S(4)
19. Z(5)
20. J(2,5,17)
21. S(5)
22. S(3)
23. J(1,1,20)
24. T(3,1)

8.Za vežbu: Napisati URM program koji izračunava funkciju $f(x) = 2 * x + 2 * y + 2 * z$

Rešenje se svodi na primenu programa za proizvod i zbir koji su nam uveliko poznati.

Ideja: primetimo da je $2 * x + 2 * y + 2 * z = 2 * (x + y + z)$ – sveli smo na jedan proizvod umesto tri, što znatno pojednostavljuje program jer znamo da je proizvod komplikovaniji od zbira za realizaciju preko URM instrukcija.

9.Napisati URM program koji izračunava funkciju $f(x) = 2^x$

Predstavićemo operaciju stepenovanja preko uzastopne primene operacije proizvoda:

$$2^x = 2 * 2 * 2 * \ldots * 2 = 2^{x-1} * 2$$

Ideja je da postupno računamo vrednosti $2^0, 2^1, 2^2, \ldots$ Primetimo da svaku narednu vrednost dobijamo množenjem prethodne sa 2 – postupak ponavljamo upravo x puta.

Početna konfiguracija – argument je samo x, broj 2 moramo sami da napravimo

```
------------------------
| x |
------------------------
```

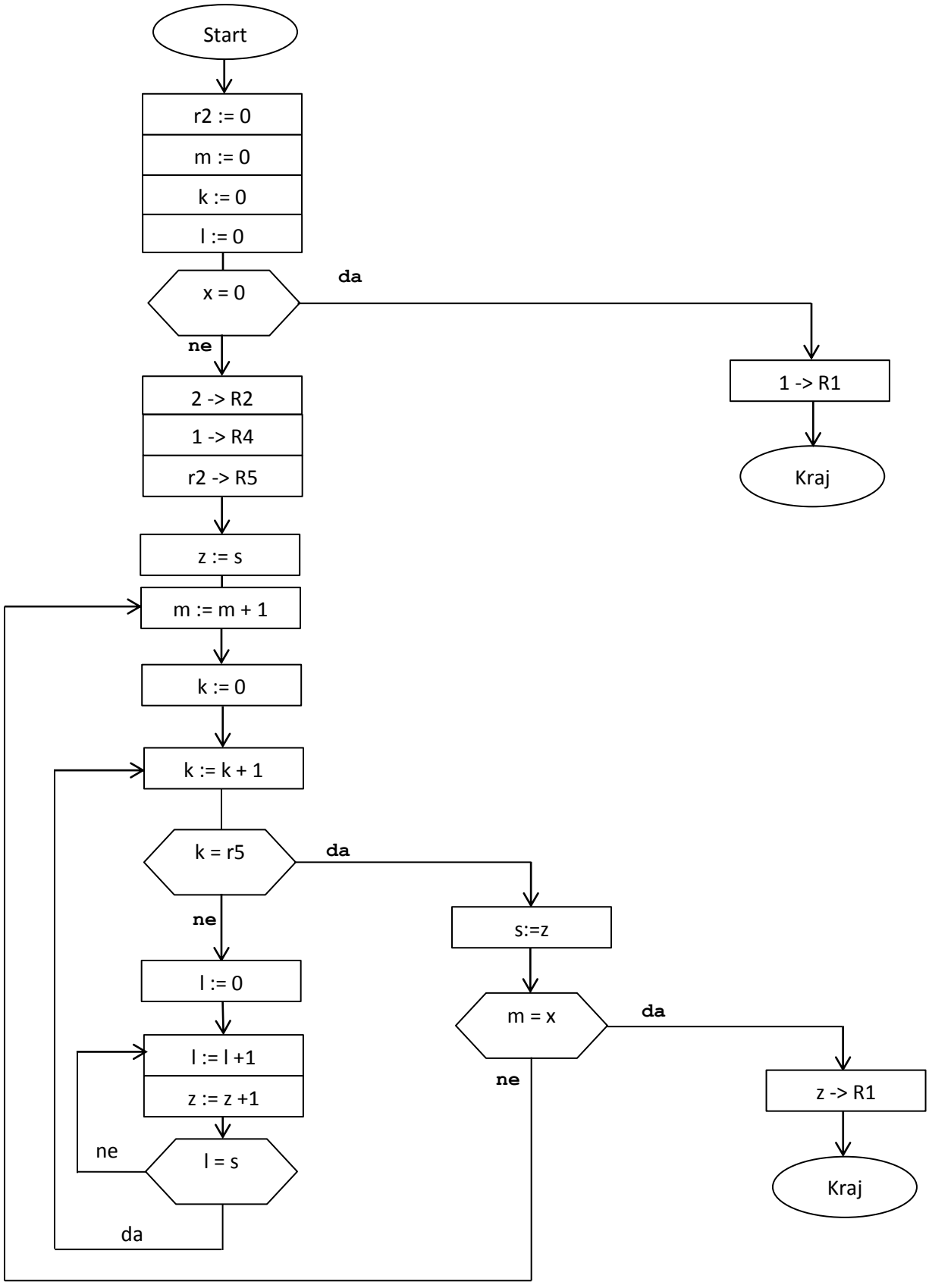
Konfiguracija u toku rada
Registri R4 – R8 služe za realizaciju proizvoda i upravo odgovaraju registrima koje smo koristili u programu za proizvod
– m je brojač za x, dokle smo stigli sa pravljenjem 2^x
- s je oblika 2^i, tj. uzima vrednosti $2^0, 2^1, 2^2, \ldots$ redom
- u svakom koraku množimo tekuću vrednost s sa 2
- u z se smešta s*2, i odatle prebacujemo ponovo u s kao njegovu novu vrednost jer ponavljamo postupak opisan na početku zadatka
- k i l su brojači potrebni za množenje, k brojač koji uzima vrednosti 0,1,2, a l ide od 0 do tekuće vrednosti za s

x	2	m	s	2	z	k	l

URM program:

1. Z(2)
2. Z(3)
3. Z(7)
4. Z(8)
5. J(1,3,21)
6. S(2)
7. S(2)
8. Z(4)
9. S(4)
10. T(2,5)
11. T(4,6)
12. S(3)
13. Z(7)
14. S(7)
15. J(7,5,24)
16. Z(8)
17. S(8)
18. S(6)
19. J(8,4,14)
20. J(1,1,17)
21. Z(1)
22. S(1)
23. J(1,1,100)
24. T(6,4)
25. J(3,1,27)
26. J(1,1,12)
27. T(6,1)

Dijagram toka



10.Napisati URM program koji pravi broj 1000.

Primetimo da je $\mathbf{1000 = 10^3}$ pa možemo iskoristiti prethodni zadatak – umesto 2 treba napraviti broj 10, a za x se uzima konkretna vrednost, tj. broj 3 – nije data funkcija, pa nema ulaznih argumenata, sve brojeve pravimo sami.

Početna konfiguracija – prazna traka

```
------------------------
|
------------------------
```

Konfiguracija u toku rada
Registri R4 – R8 služe za realizaciju proizvoda i upravo odgovaraju registrima koje smo koristili u programu za proizvod
– m je brojač za 3, dokle smo stigli sa pravljenjem 10^3
- s je oblika 10^i , tj. uzima vrednosti $10^0, 10^1, \ldots$ redom
- u svakom koraku množimo tekuću vrednost s sa 10
- u z se smešta s*10, i odatle prebacujemo ponovo u s kao njegovu novu vrednost jer ponavljamo postupak opisan na početku zadatka
- k i l su brojači potrebni za množenje, k brojač koji uzima vrednosti 0,1,…,10, a l ide od 0 do tekuće vrednosti za s

```
-------------------------------------------
| 3 | 10 | m | s | 10 | z | k | l |
-------------------------------------------
```

URM program:

```
1. Z(1)
2. S(1)
3. S(1)
4. S(1)
5. Z(2)
6. S(2)
7. S(2)
8. S(2)
9. S(2)
10.S(2)
11.S(2)
12.S(2)
13.S(2)
14.S(2)
15.S(2)
16.Z(3)
17.Z(7)
18.Z(8)
19.J(1,3,35)
20.S(2)
21.S(2)
22.Z(4)
23.S(4)
24.T(2,5)
25.T(4,6)
```

```
26.S(3)
27.Z(7)
28.S(7)
29.J(7,5,38)
30.Z(8)
31.S(8)
32.S(6)
33.J(4,8,28)
34.J(1,1,31)
35.Z(1)
36.S(1)
37.J(1,1,100)
38.T(6,4)
39.J(3,1,41)
40.J(1,1,26)
41.T(6,1)
```

11.Napisati URM program koji pravi broj 2198.

Iskoristiti algoritam za stepenovanje i instrukciju sledbenik
$\mathbf{2198 = 2197 + 1 = 13^3 + 1}$

12. Napisati URM program koji izračunava funkciju $f(x,y) = \begin{cases} x - 10, & ako\ je\ x > 10 \\ 0, & ako\ je\ x <= y \end{cases}$

Iskoristiti 3.zadatak – y je ovde jednako 10, ne dobija se na početku kao ulazni argument već broj 10 pravimo sami na objašnjen način.