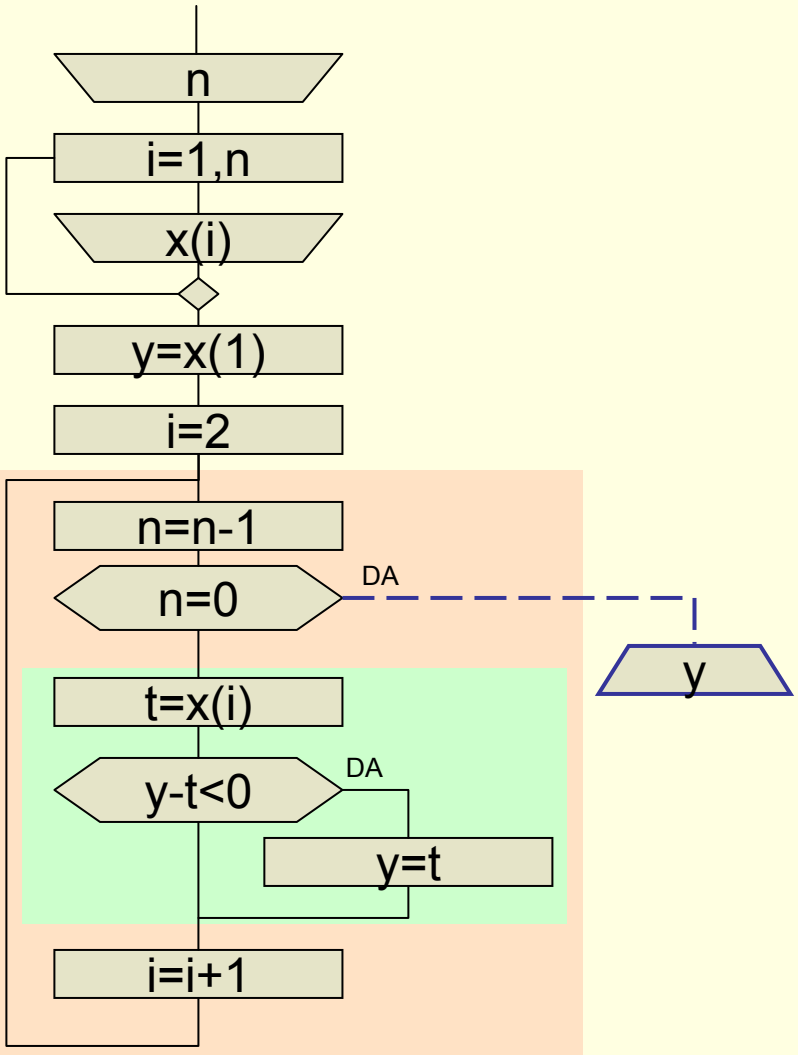
Načini adresiranja

Rad sa nizovima

Programska modifikacija - Minimum *n* (*n* ≤ 10) realnih brojeva




```
100 DR 4.0,2.1,3.4,5.2 ;  niz x
104 DM 6               ;  prazna mesta za niz
110 DC 4,1             ;  n, 1
112 DR 1               ;  y
113 DM 1               ;  lokacija za restauraciju
114 DM 1               ;  pomocna promenljiva, t
115 MUA 123            ;  naredba → AK
116 AUM 113            ;  priprema za restauracija
117 MUA 100            ;  x(1) → AK
118 AUM 112            ;  y = x(1)
119 MUA 110            ;  n → AK
120 ODUF 111           ;  S(AK) – 1
121 AUM 110            ;  n = n – 1
122 NUS 134            ;  n = 0 ? → kraj
123 MUA 100            ;  x(i) → AK
124 AUM 114            ;  t = x(i)
125 MUA 112            ;  y → AK
126 ODU 114            ;  S(AK) – t
127 NES 130            ;  y – t < 0?
128 MUA 114            ;  t → AK
129 AUM 112            ;  y = t
130 MUA 123            ;  naredba → AK
131 SABF 111           ;  S(AK) + 1
132 AUM 123            ;  programska modifikacija
133 BES 119
134 MUA 113
135 AUM 123            ;  programska restauracija
136 ZAR
```

Programska modifikacija

- Primer ilustruje modifikaciju adresnog dela instrukcije
- Moguće je modifikovati i operacioni deo instrukcije
- Nedostaci programske modifikacije
  - Teško je testirati programe jer se tokom izvršavanja programa menjaju i podaci i program
  - Program prestaje da bude fiksni deo kojim se definiše proces, već ulazi status vektor programa, što komplikuje rad računara u višeprogramskom režimu
  - Povećana je dužina programa jer su potrebne dodatne instrukcije za čuvanje početnih instrukcija, modifikaciju i restauraciju
- Zbog dobrih osobina modifikacije (rad sa nizovima), programska modifikacija se zamenjuje *automatskom modifikacijom* (realizuje se hardverski) koja se odnosi na adresni deo instrukcije - **adresiranje**

Direktno adresiranje

- U adresnom delu instrukcije nalazi se adresa podatka koji učestvuje u izvođenju operacije – **argument instrukcije**
- Ako je *a* adresa u adresnom delu instrukcije tada je argument instrukcije *S(a)*
- Ovakav način adesiranje se zove **direktno adresiranje**, adresa *a* je **direktna (apsolutna) adresa**
- Program napisani sa direktnim adresama zovu se **apsolutni programi**, jer moraju, pre izvršavanja, biti unete adrese operativne memorije
- Ovo pretstavlja veliko ograničenje za programe, jer bi svaki program zahtevao fikasn deo memorije i ne bi mogao da se izvršava ako je njegov deo memorije prethodno zauzet
- Zato se apsolutni programi koriste samo za neke sistemske programe

Direktno adresiranje

- Uvođenjem načina adresiranja mora se uvesti i razlika između
  - adrese u adresnom delu instrukcije (koja će biti modifikovana) – **početna adresa *p*** i
  - adrese koja će biti adresa argumenta instrukcije – **izvršna (efektivna) adresa *e***
- Način dobijanja efektivne adrese od početne adrese je određen načinom adresiranja
- U svakom slučaju

$$e=f(p,p_1,p_2,...)$$

  gde su $p_1,p_2,...$ veličine koje se nalaze u određenim registrima
- Kod direktnog adresiranja

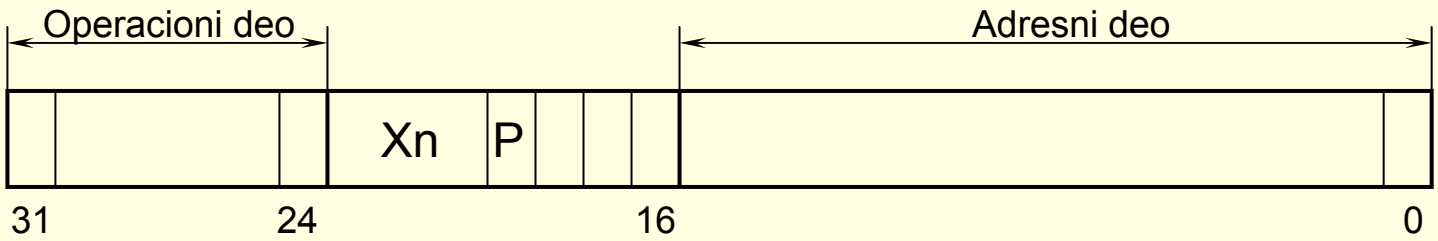
$$e=p$$

Indeksno adresiranje

- **Indeks registar** sadrži vrednost za koju se uvećava ili umanjuje početna adresa
- Računar može imati više indeks registara i u simboličkom zapisu će biti označeni sa *X1, X2, ..., X15*
- U slučaju indeksnog adresiranja

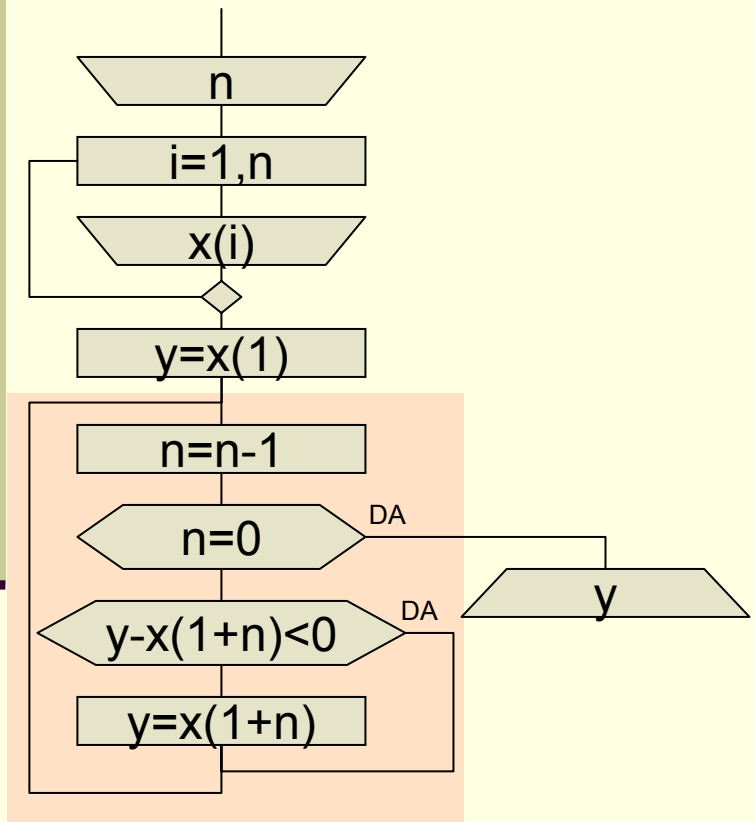
$$e=p+S(Xn),\ \ n\in\{1,2,\ldots,15\}$$

- U strukturi instrukcije mora se nalaziti adresa indeks registra i oznaka indeksnog adresiranja
- Sadržaj registra instrukcija je



Indeksno adresiranje – primer 1

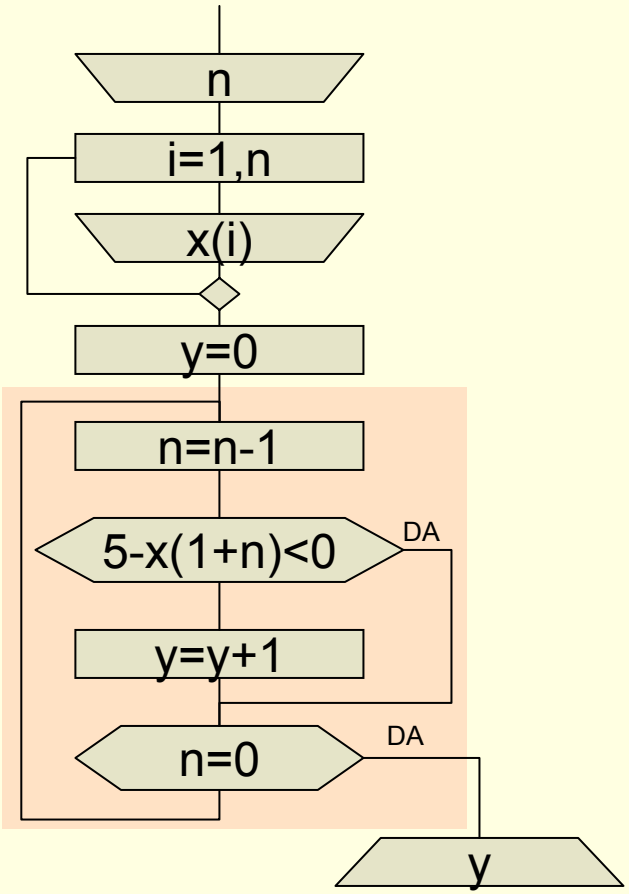
Napisati algoritam i program koji za niz od *n* (*n*≤10) realnih brojeva određuje minimalni element




```
100 DR 4.0,2.1,3.4,5.2      ;  niz x
104 DM 6
110 DC 4,1                  ;  n,1
112 DM  1                   ;  y
113 MUA 100                 ;  x(1) → AK
114 AUM 112                 ;  y = x(1)
115 MUA 110                 ;  n → S(AK)
116 ODUF 111                ; S(AK) - 1
117 AUM 110                 ;  n = n - 1
118 NUS 126                 ;  n = 0?  → kraj
119 PIR X1,110              ;  n → X1
120 MUA 112                 ;  y → AK
121 ODUF X1,P,100           ; S(AK) - x(1+n)
122 NES 125                 ;  y < x(1+n)?
123 MUA X1,P,100            ;  x(1+n) → AK
124 AUM 112                 ;  y = x(1+n)
125 BES 115
126 ZAR
```

Indeksno adresiranje – primer 2

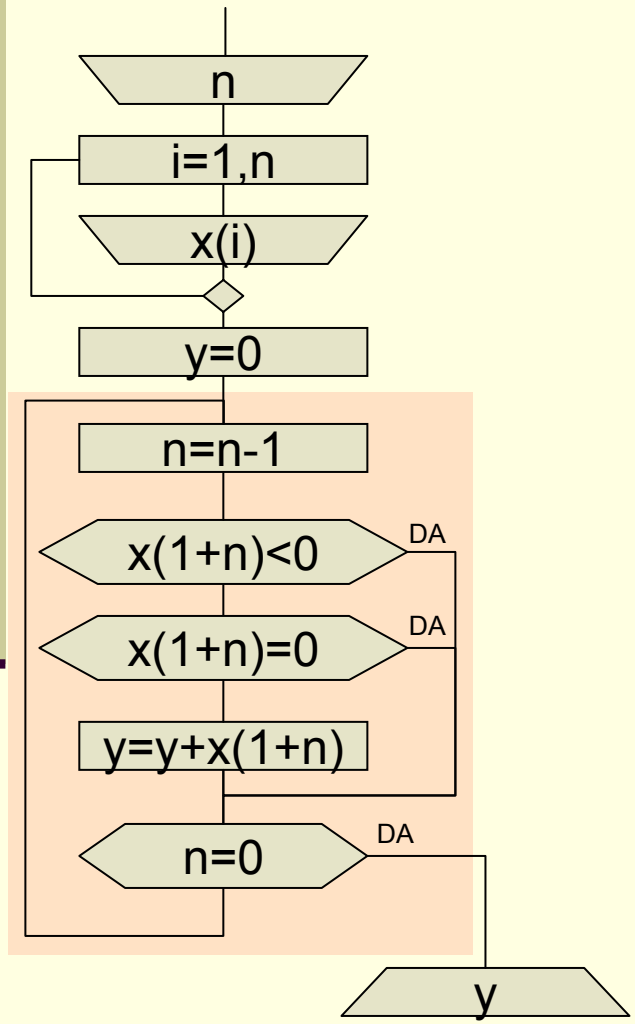
Napisati algoritam i program koji za niz od *n* (*n*≤10) realnih brojeva određuje broj elemenata koji su manji ili jednaki 5




```
100 DR 4.0,8.1,6.4,5,3.2,2 ;  niz x
106 DM 4
110 DC 6,1,5               ;  n,1,5
113 DC 0                   ;  y
114 MUA 110                ;  n → AK
115 ODUF 111               ; S(AK) – 1
116 AUM 110                ;  n = n – 1
117 PIR X1,110             ;  n → X1
118 MUA 112                ;  5 → AK
119 ODUF X1,P,100          ; S(AK) – x(1+n)
120 NES 124                ;  x(1+n) > 5?
121 MUA 113                ;  y → AK
122 SABF 111               ; S(AK) + 1
123 AUM 113                ;  y = y + 1
124 MUA 110                ;  n → AK
125 NUS 127                ;  n = 0? → kraj
126 BES 114
127 ZAR
```

Indeksno adresiranje – primer 3

Napisati algoritam i program koji za niz od *n* (*n*≤10) celih brojeva određuje sumu pozitivnih elemenata niza

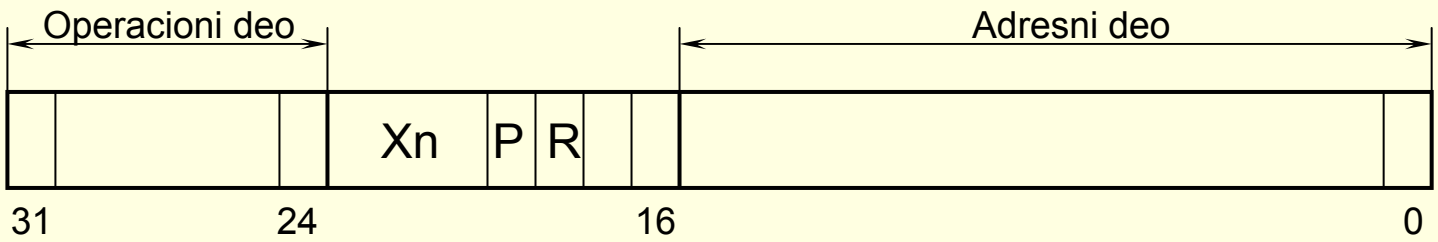



```
100 DC 4,-8,-6,5,3,2        ;  niz x
106 DM 4
110 DC 6,1                  ;  n,1
113 DC 0                    ;  y
114 MUA 110                 ;  n → AK
115 ODUF 111                ; S(AK) – 1
116 AUM 110                 ;  n = n – 1
117 PIR X1,110              ;  n → X1
118 MUA X1,P,100            ;  x(1+n) → AK
119 NES 124                 ;  x(1+n) < 0?
120 NUS 124                 ;  x(1+n) = 0?
121 MUA 113                 ;  y → AK
122 SABF X1,P,100           ; S(AK) + x(1+n)
123 AUM 113                 ;  y = y + x(1+n)
124 MUA 110                 ;  n → AK
125 NUS 127                 ;  n = 0? → kraj
126 BES 114
127 ZAR
```

Relativno adresiranje

- Adresni deo instrukcije, osim sadržajem indeks registra, može biti modifikovan i brojačem instrukcija
- Brojač instrukcija ukazuje na adresu instrukcije koja treba da se izvrši, pa uvećanje adresnog dela instrukcije, za njegov sadržaj, ima efekat dobijanja izvršne adrese koja je za vrednost početne adrese uvećana ili umanjena u odnosu na adresu tekuće instrukcije, tj.

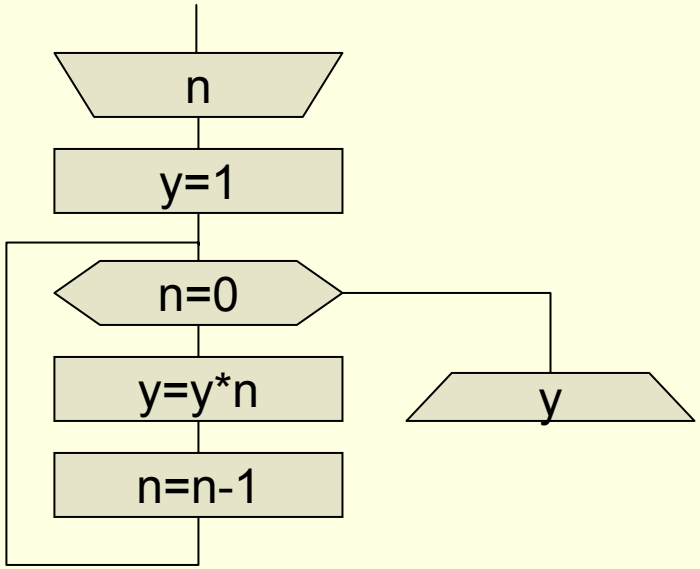
$$e=p+S(BI)$$



Relativno adresiranje - primer

Napisati algoritam i program koji za ceo broj *n* određuje *n!*

$$n!=n\cdot(n-1)\cdot\ldots\cdot 2\cdot 1$$

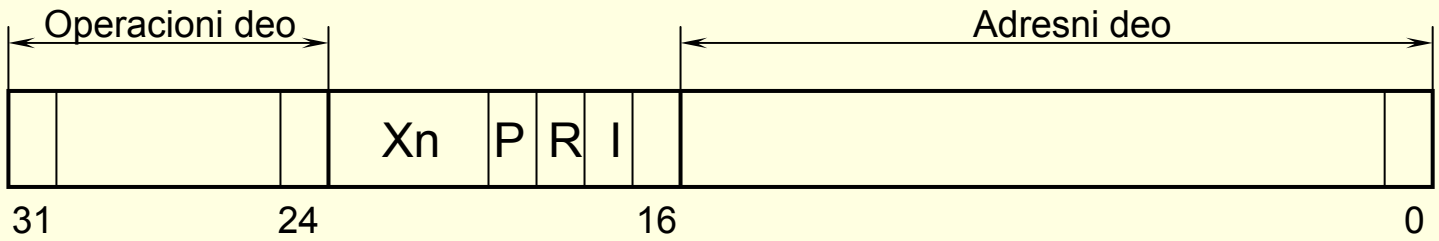



```
0 DC 4          ;  n
1 DC 1          ;  y
2 DC 1          ;  1
3 MUA R,-3      ;  n → AK
4 NUS R,7       ;  n = 0 ? → kraj
5 MNO R,-4      ;  S(AK) * y
6 AUM R,-5      ;  y = y * n
7 MUA R,-7      ;  n → AK
8 ODUF R,-6     ;  S(AK) – 1
9 AUM R,-9      ;  n = n – 1
10 BES R,-7
11 ZAR
```

Indirektno adresiranje

- Adresa argumenta se nalazi u posebnom memorijskom registru, a početna adresa u adresnom delu instrukcije, tj.

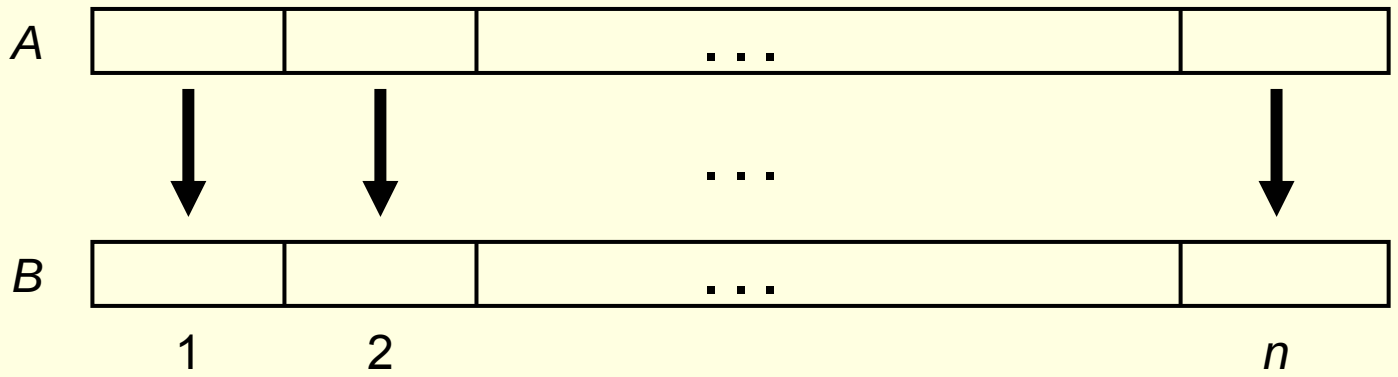
$$e=S(p)$$



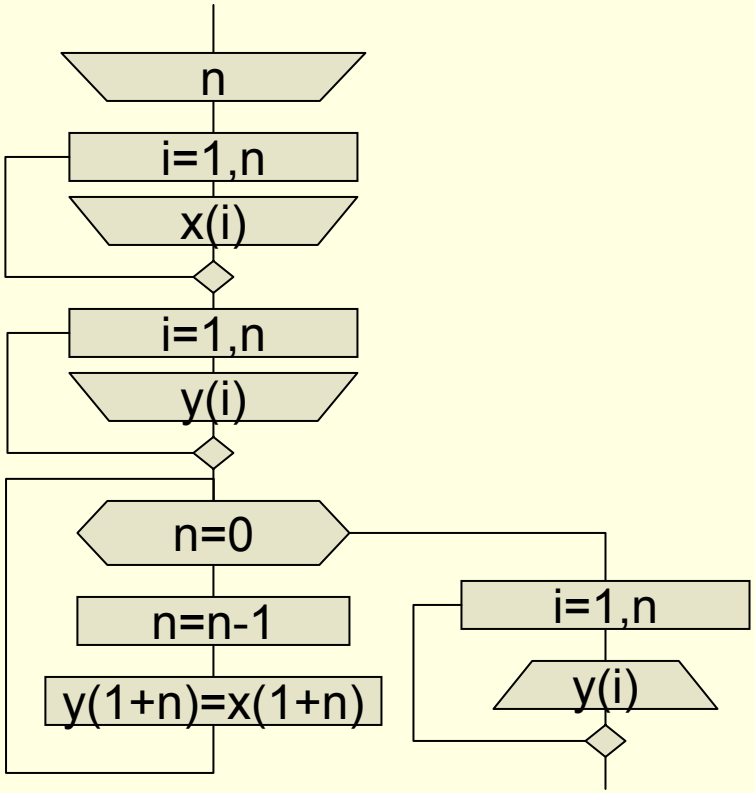
- Kod ovakvog adresiranja vrši se zamena adresnog dela instrukcije novim sadržajem iz memorije, pri čemu se može promeniti i način adresiranja, jer se u registru instrukcija ne menja samo oprecioni deo
- Nov sadržaj u registru instrukcija može definisati neku od vrsta modifikacija, uključujući i novo indirektno adresiranje – **indirektno adresiranje po dubini**

Indirektno adresiranje - primer

- Napisati algoritam i program koji prenosi sadržaje $n \leq 10$ memorijskih registara iz zone sa adresom *A* u zonu sa adresom *B*.



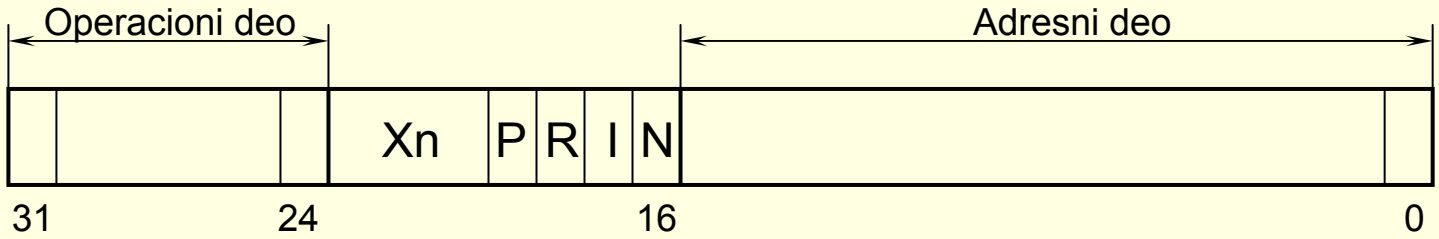
Indirektno adresiranje - primer




```
100 DC 00180000H    ; X1, P, A
101 DC 00180000AH   ; X1, P, B
102 DC 8            ; n
103 DC 1            ; 1
104 MUA 102         ; n → AK
105 NUS 112         ; n=0? – kraj
106 ODUF 103        ; S(AK) – 1
107 AUM 102         ; n=n – 1
108 PIR X1,102      ; n → X1
109 MUA I,100       ; MUA X1,P,0
110 AUM I.101       ; AUM X1,P,10
111 BES 104
112 ZAR
```

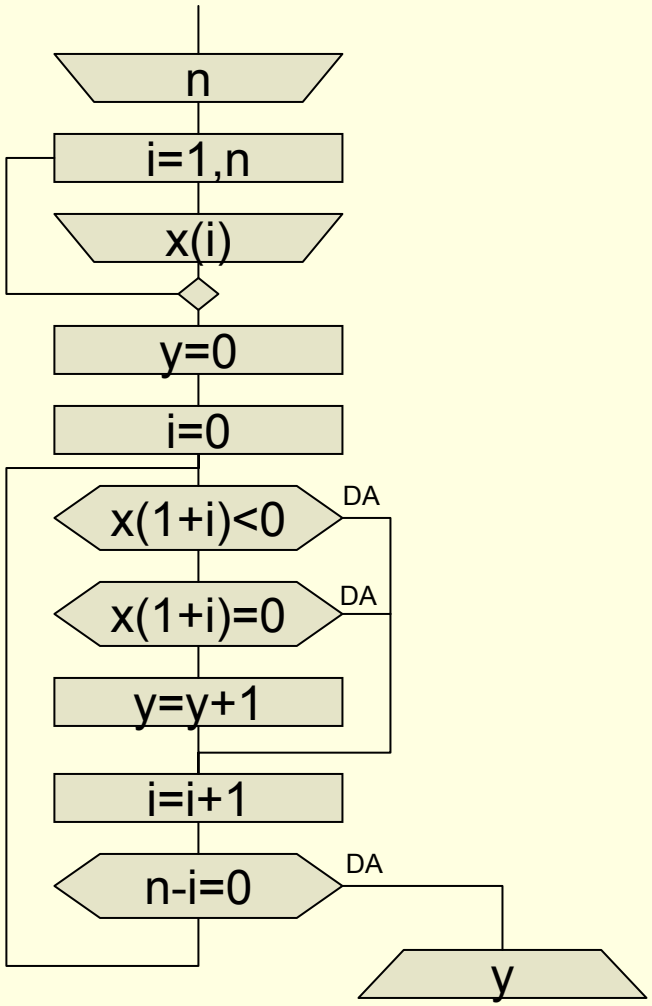
Neposredno adresiranje

- Podaci koji učestvuju kao argumenti, pri izvršavanju instrukcije, najčešće se dobijaju iz drugih registara, pa je nophodno raspolagati tim adresama
- Kako je i adresa podatak, ona može postati argument instrukcije – **neposredno adresiranje**
- Argument instrukcije se koristi u fazi izvršenja, pa se izvršna adresa formirana u fazi pripreme koristi kao argument instrukcije
- Neposredno adresiranje ima smisla primeniti na instrukcije SABF, ODUF, MNOF, DELF, MUA, PIR



Neposredno adresiranje – primer 1

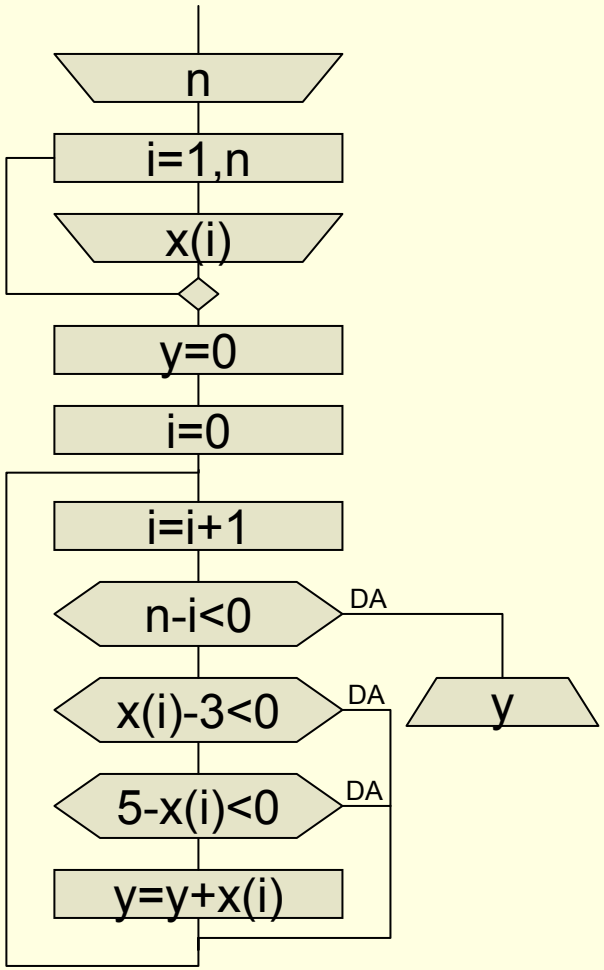
Napisati algoritam i program koji za niz od *n* (*n*≤10) celih brojeva određuje broj pozitivnih elemenata niza




```
100 DM 1                ;  y
101 DC 4,-8,-6,5,3,2    ;  niz x
107 DM 4
111 DC 6                ;  n
112 MUA N,0             ;  0 → AK
113 AUM 100             ;  y = 0
114 PIR X1,N,0          ;  0 → X1, i = 0
115 MUA X1,P,101        ;  x(i) → AK
116 NES 121             ;  x(i) < 0?
117 NUS 121             ;  x(i) = 0?
118 MUA 100             ;  y → AK
119 SABF N,1            ;  S(AK) + 1
120 AUM 100             ;  y = y + 1
121 PIR X1,P,N,1        ;  S(X1) + 1 → X1,  i=i+1
122 MUA 111             ;  n → AK
123 ODUF X1,P,N,0       ;  S(AK) – S(X1)
124 NUS 126             ;  n – i = 0 ? → kraj
125 BES 115
126 ZAR
```

Neposredno adresiranje – primer 2

Napisati algoritam i program koji za niz od n ($n \leq 10$) realnih brojeva određuje sumu elemenata niza za koje važi $3 \leq x_i \leq 5$.

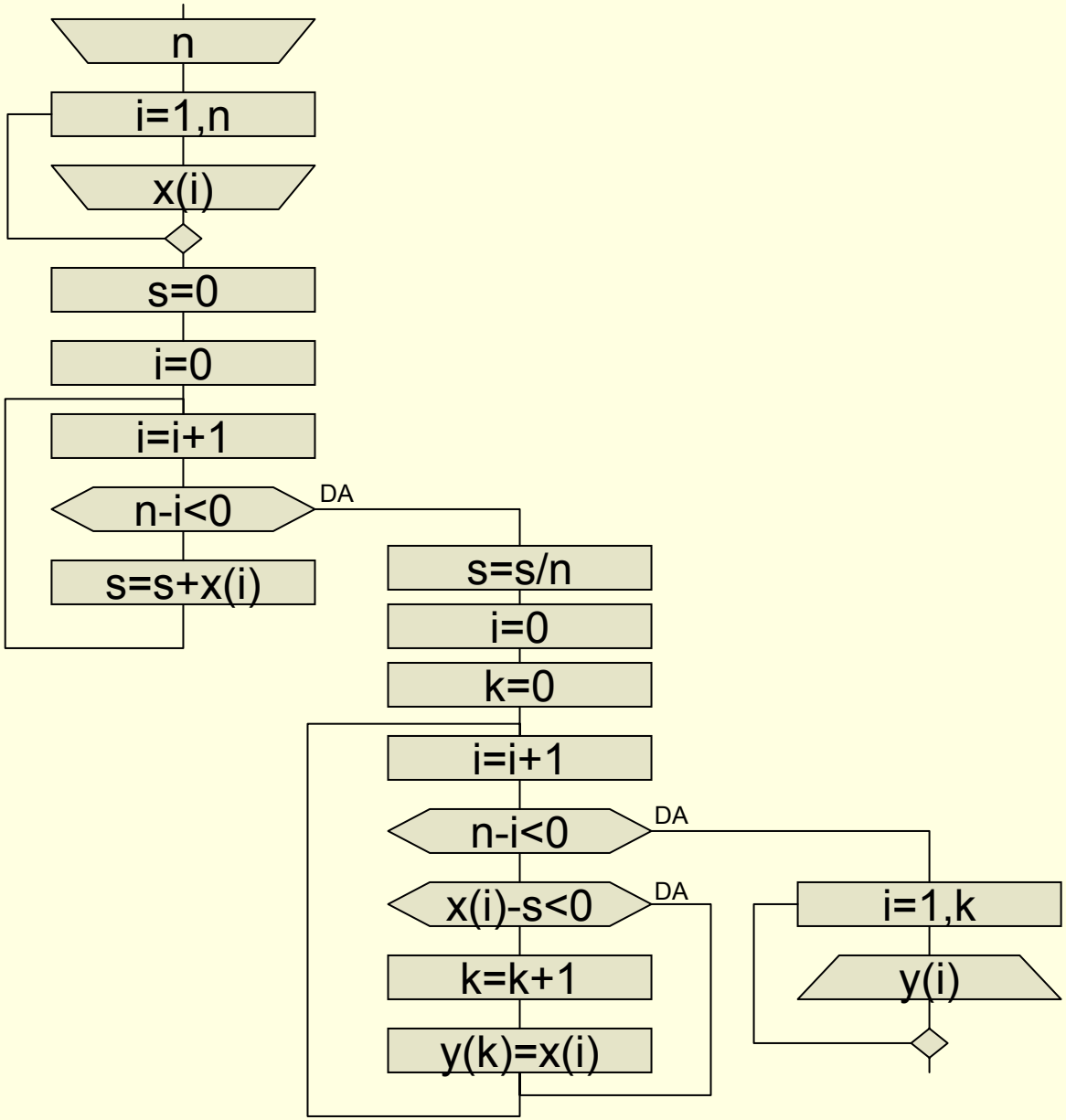



```
100 DM 1                    ; y
101 DC 4.2,6.3,3.1,2.5      ; niz x
105 DM 6
111 DC 4                    ; n
112 MUA N,0                 ; 0 → AK
113 AUM 100                 ; y = 0
114 PIR X1,N,0              ; 0 → X1, i = 0
115 PIR X1,P,N,1            ; S(X1) + 1 → X1,  i=i+1
116 MUA 111                 ; n → AK
117 ODUF X1,P,N,0           ; S(AK) – S(X1)
118 NUS 129                 ; n – i < 0 ? → kraj
119 MUA X1,P,100            ; x(i) → AK
120 ODU N,3                 ; S(AK) – 3
121 NES 128                 ; 3 ≤ x(i) ?
122 MUA N,5                 ; 5 → AK
123 ODU X1,P,100            ; S(AK) – x(i)
124 NES 128                 ; x(i) ≤ 5 ?
125 MUA 100                 ; y → AK
126 SAB X1,P,N,0            ; S(AK) + x(i)
127 AUM 100                 ; y = y + x(i)
128 BES 115
129 ZAR
```

Neposredno adresiranje – primer 3

Napisati algoritam i program koji za niz *x* od *n* (*n*≤10) realnih brojeva formira nov niz *y* čiji su elementi oni članovi niza *x* koji su veći ili jednaki srednjoj vrednosti niza *x*

Neposredno adresiranje – primer 3



Neposredno adresiranje – primer 3

```
100 DM 1              ; s
101 DR 4,8,6,5,3      ; niz x
107 DM 5
111 DM 10             ; niz y
121 DC 5              ; n
122 MUA N,0           ; 0 → AK
123 AUM 100           ; s = 0
124 PIR X1,N,0        ; 0 → X1, i = 0
125 PIR X1,P,N,1      ; S(X1)+1→X1,i=i+1
126 MUA 121           ; n → AK
127 ODUF X1,P,N,0     ; S(AK) – S(X1)
128 NES 133           ; n – i < 0 ?
129 MUA 100           ; s → AK
130 SAB X1,P,100      ; S(AK) + x(i)
131 AUM 100           ; s = s + x(i)
132 BES 125
133 MUA 100           ; s → AK
134 DEL 121           ; S(AK) / n
135 AUM 100           ; s = s / n
136 PIR X1,N,0        ; 0 → X1, i = 0
137 PIR X2,N,0        ; 0 → X2, k = 0
138 PIR X1,P,N,1      ; S(X1)+1→X1,i=i+1
139 MUA 121           ; n → AK
140 ODUF X1,P,N,0     ; S(AK) – S(X1)
141 NES 149           ; n – i < 0 ?
142 MUA X1,P,100      ; x(i) → AK
143 ODU 100           ; S(AK) – s
144 NES 148           ; x(i) < s
145 PIR X2,P,N,1      ; S(X2)+1→X2,k=k+1
146 MUA X1,P,100      ; x(i) → AK
147 AUM X2,P,110      ; y(k) = x(i)
148 BES 138
149 ZAR
```