

Vektorski računari



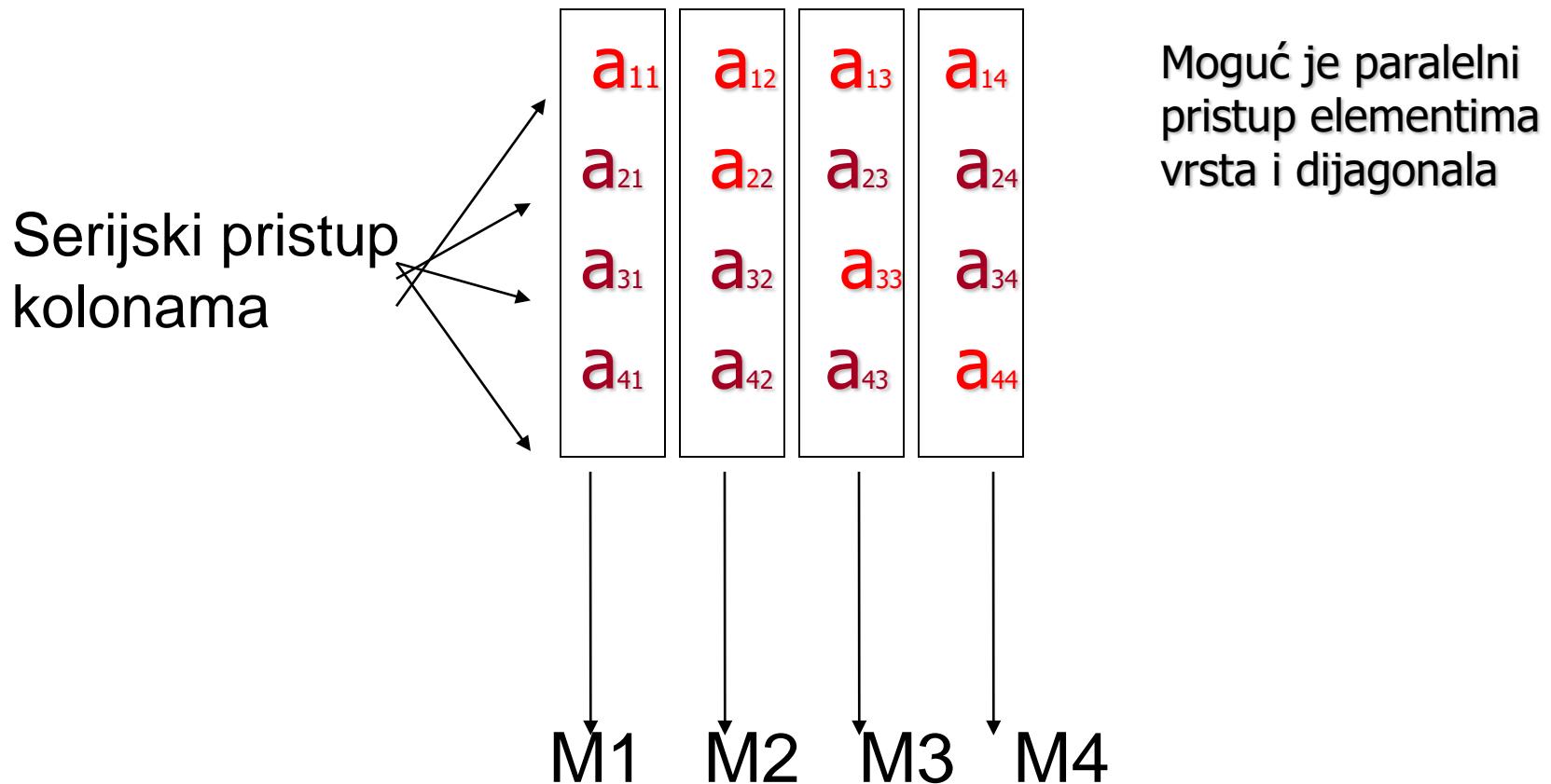
Smeštanje podataka i određivanje broja
memorijskih banaka

Smeštanje podataka

- * Jedan od ključnih faktora koji utiče na vreme izvršenja programa na paralelnom procesoru je latentnost memorijskog sistema
 - vreme od trenutka izdavanja zahteva za pribavljanjem podatka do trenutaka kada on postane dostupan
 - da bi se problem rešio koristi se paralelne memoriske banke
 - način smeštanja podataka igra važnu ulogu, jer može smanjiti vreme pristupa elementima polja
- * Pošto je osnovna struktura podataka koja se koristi kod vektorskih računara polje, način smeštanja elemenata polja u memoriske module može bitno uticati na efikasnost vektorskog izračunavanja tako što će smanjiti vreme pristupa elementima polja

Smeštanje podataka – primer

Razmotrimo moguće načine smeštanja elemenata matrice A dimenzija 4×4 u mem. banke,

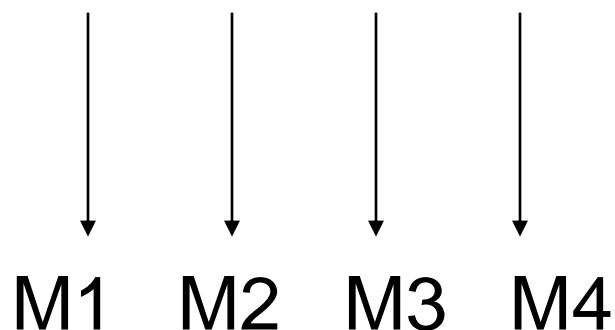


Primer - nastavak

Serijski pristup
vrstama

a_{11}	a_{21}	a_{31}	a_{41}
a_{12}	a_{22}	a_{32}	a_{42}
a_{13}	a_{23}	a_{33}	a_{43}
a_{14}	a_{24}	a_{34}	a_{44}

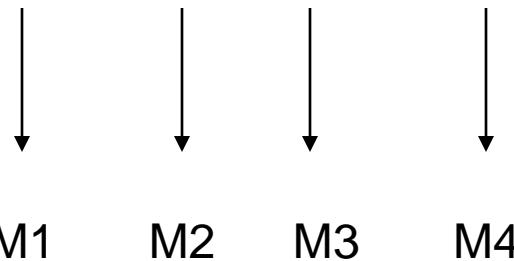
Paralelni pristup po
kolonama i
dijagonalama



Primer - nastavak

Paralelni pristup
elementima kolona i
vrsta

a_{11}	a_{12}	a_{13}	a_{14}
a_{24}	a_{21}	a_{22}	a_{23}
a_{33}	a_{34}	a_{31}	a_{32}
a_{42}	a_{43}	a_{44}	a_{41}

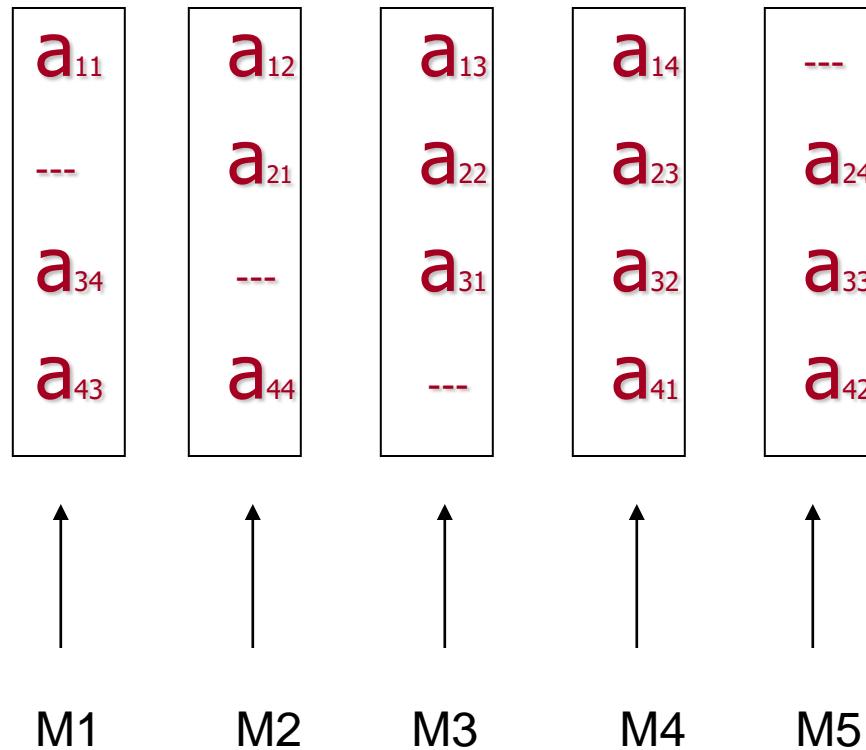


Konflikti kod pristupa dijagonalnim elementima!

Primer - nastavak

Paralelni pristup kolonama/vrstama/dijagonalama

(5 memorijskih banaka)



- Teorijski, da bi se obezedio pristup proizvoljnoj strukturi dvodimenzionalnog polja, potrebno je da broj memorijskih banaka bude veći od dimenzije matrice

Odredjivanje broja memorijskih banaka

- Primer: množenje matrica

```
for i =1, 100  
for j =1, 100  
A(i,j) = 0.0  
do 10 k =1, 100  
A(i,j) = A(i,j) +B(i,k) * C(k,j)  
endfor{i,j}
```

- Petlju po indeksnoj promenljivoj k je moguće vektorizovati tako što bi se obavilo množenje vrsta matrice B kolonama matrice C.

```
for i =1, 100  
for j =1, 100  
A(i,j) = 0.0  
A(i,j) = A(i,j) +B(i,1:100) * C(1:100,j)  
endfor{i,j}
```

- Da bi ustanovili efikasnost vektorizacije moramo razmotriti kako su susedni elementi u matricama B i C adresirani.

Odredjivanje broja memorijskih banaka (nast.)

* Kada se vrši smeštanje elemenata dvodimenzionalnog polja u memoriju vrši se linearizacija .

- Ako se u suksesivne mem. lokacije smeštaju elementi kolona, znači da u primeru množenja matrica elementi matrice B kojima treba pristupiti u jednoj iteraciji nisu na suksesivnim memorijskim adresama već su udaljeni medjusobno za

broj_vrsta x dužina_reči_u_bajtovima

- Razmak izmedju susednih elemenata kojima treba pristupiti suksesivno zove se vektorski korak ili pomeraj (vector stride).
- U primeru množenja matrica, vektorski korak za matricu C je 1, dok je za B jednak 100.
- Kada se vektor pribavi u vektorski registar on se ponaša kao da ima logički susedne elemente.

Odredjivanje broja memorijskih banaka (nast.)

* Vektorski procesor može da upravlja vektorskim korakom > 1 pomoću vektorskih LOAD/ STORE operacija.

- Ova sposobnost da pristupa nesekvencijalnim memorijskim lokacijama i da ih prevodi u strukturu sa logički susednim elementima je jedna od značajnih prednosti vektorskih procesora nad keš baziranim procesorima.
 - Vektorski korak kao i startna adresa vektora mogu se zapamtiti u neki od registara opšte namene.
 - Zatim se instrukcija tipa LVWS (load_vector_with_stride -- napuni vektor sa korakom) može upotrebiti da se pribavi vektor u vektorski register (slično i za STORE naredbu SVWS).
 - Kod nekih vektorskih procesora LOAD i STORE uvek imaju vrednost koraka zapamćenu u registru, pa nema posebnih LOAD i STORE instrukcija

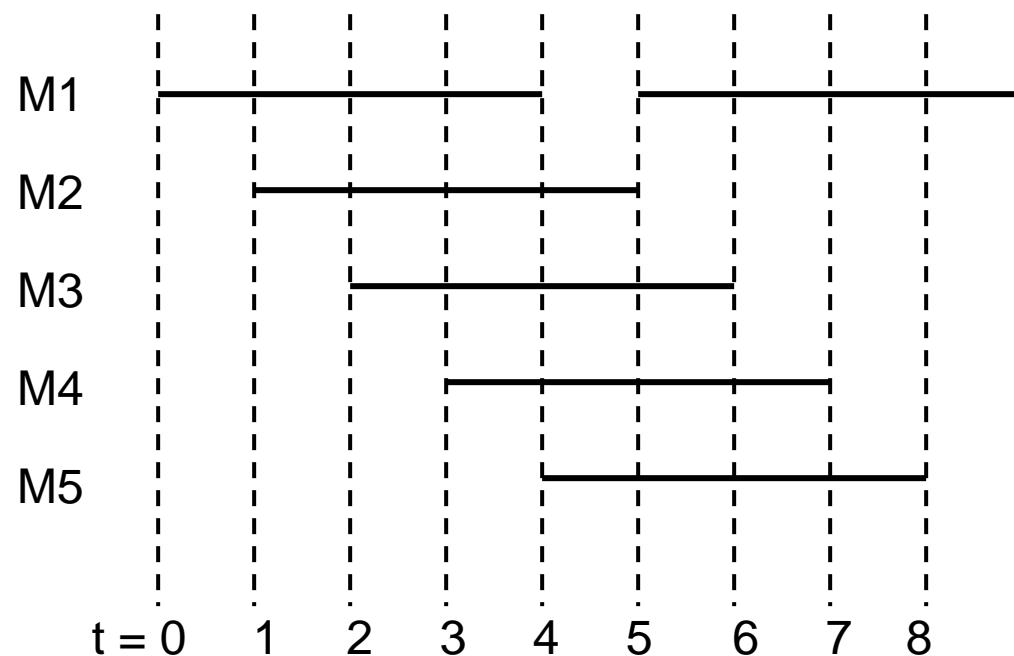
Odredjivanje broja memorijskih banaka (nast.)

* Komplikacije u memorijskom sistemu mogu da nastupe kad treba podržati korake > 1 .

- U opštem slučaju da bi se mogao pribaviti jedan element vektora po klok ciklusu, broj memorijskih banaka mora biti veći od latentnosti memoriskog sistema (vreme pristupa memoriji).
- Kada postoji korak > 1 može se desiti da se zahteva pristup istoj memorijskoj baci većom brzinom od brzine memoriskog ciklusa.
 - U takvim slučajevima jedan zahtev mora biti zakašnjen zbog postojanja konflikta.

A1	A2	A3	A4	A5
A6	A7	A8	A9	A10
M1	M2	M3	M4	M5

Latentnost =4



Odredjivanje broja memorijskih banaka (nast.)

* Konflikt kod pristupa memorijskoj banci nastupa ako važi sledeći uslov

$$\frac{NZS(vektorski_korak, Broj_memorijskih_banaka)}{vektorski_korak} \leq latentnost_memorijskog_sistema$$

- NZS je najmanji zajednički sadržalac.

* **PRIMER.** Prepostavimo da imamo 16 memorijskih banaka sa latentnošću od 12 clk ciklusa. Koliko vremena će biti potrebno da se napuni 64-elementni vektor ako se elementi koji se pribavljuju nalaze na medjusobnom rastojanju

- 1 (tj. vektorski korak je 1)
- 32 (tj. vektorski korak je 32)

Odredjivanje broja memorijskih banaka -primer

* REŠENJE:

- a) Pošto je broj memorijskih banaka (16) veći od latentnosti memorijskog sistema (12) za korak 1 imaćemo da je vreme potrebno da se pribave 64 elementa

$$* 12 + 63 = 75 \text{ clk ciklusa ili } 1.2 \text{ clk/element}$$

• b)

$$\frac{NZS(32,16)}{32} = \frac{32}{32} = 1 < 16$$

- Najgori mogući vektorski korak je umnožak od broja memorijskih banaka, kao u ovom primeru.

➤ U ovakvim situacijama svi elementi kojima treba pristupiti se nalaze u istoj memorijskoj banci pa je latentnost memorijskog sistema vidljiva za svaki element koji treba pribaviti umesto samo jednom kod prvog pristupa kada je korak 1.

➤ U našem primeru to dovodi do latentnosti od 12 clk ciklusa po elementu, tj. za ceo vektor: **12 x 64 = 768 clk**

* Konflikt kod pristupa memoriji neće nastupiti ako su vektorski korak i broj memorijskih banaka uzajamno prosti brojevi i ako postoji dovoljno memorijskih banaka da ne nastupi konflikt kada je vektorski korak 1 (odnosno ako je broj banaka veći od latentnosti memorijskog sistema).