# Paralelni sistemi

predavanja: Emina Milovanović (L3)

vežbe: Natalija Stojanović (L3)

fond sati 2+2+1 (VIII sem.)

cs.elfak.ni.ac.rs/nastava/

# Način ocenjivanja

- Lab. Vežbe 20
- I kolokvijum 40 (>20)
- II kolokvijum 40 (>20)

---

- Ukupno 100

- Lab. Vežbe 20
- Pisani deo ispita 40
- Usmeni deo 40

---

- Ukupno 100

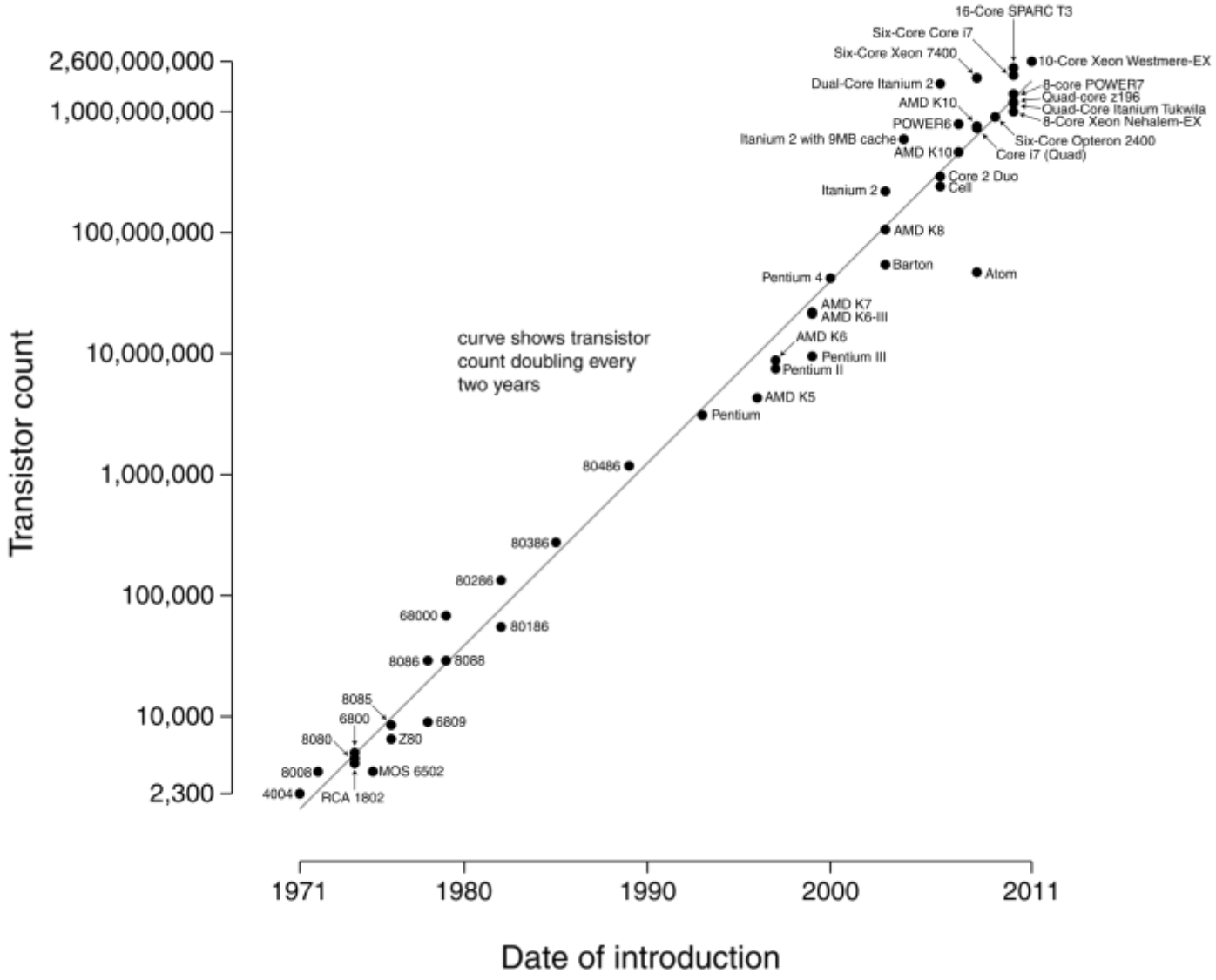
Literatura


1. prezentacije sa predavanja cs.elfak.ni.ac.rs/predavanja/
2. M. Stočev, E. Milovanović, T. Nikolić, Višeprocesorski sistemi na čipu, Elektronski fakultet Niš, 2012.

# Moore-ov zakon

- Kompjuterska tehnologija je učinila neverovatan napredak u poslednjih 60 godina od kada se pojavio prvi elektronski računar opšte namene.
  - Danas se za manje od 1000 Eur može kupiti PC koji ima bolje performanse, više memorije i više prostora na disku od računara proizvedenog 80-ih godina 20. v. Koji je koštao milione dolara.
- **Moore-ov zakon** kaže da se tokom istorije računarstva broj tranzistora na integrisanom kolu udvostručava približno svake 2 godine.
  - Zakon je dobio ime po suosnivaču Intela, Gordon E. Moore, koji je ovoaj trend opisao u svom radu koji je publikovan 1965. god.
  - Pokazalo se da je njegovo predviđanje prilično pouzdano.
- Moore-ov zakon se danas koristi da opiše trend performansi  mnogih elektronskih komponente
  - Brzina procesiranja, kapacitet memorije, pa čak i broj piksela kod digitalne kamere.
  - Sve promene su eksponencijalne.
  - Smatralo se da će ovakav trend da se nastavi sve do 2015-2020. Međutim, od 2002 taj rast je sporiji i performanse se dupliraju na svake 3 godine.

# Microprocessor Transistor Counts 1971-2011 & Moore's Law



- Brzina izračunavanja se povećava, ali i potrebe za većim brzinama izračunavanja rastu.

* Napredak je postignut zahvaljujući
  - Boljoj tehnologiji
    - Veća gustina=> manje rastojanje=> brži procesori.
    - Brži procesori => veća potrošnja energije
    - Veća potrošnja energije => veće zagrevanje
    - Veće zagrevanje => nepouzdani procesori
    - Ograničenja: brzina prostiranja svetlosti kroz vakum, stepen integracije (zbog medjusobne interakcije komponenti)
  - Promeni u arhitekturi računara
    - Korišćenje protočnosti, više specijalizovanih funkcionalnih jedinica (koprocesora)
  - Korišćenju više procesora (paralelizma)
    - Umesto gradnje bržih mikroprocesora, staviti više procesora na jedan čip
    - Sistem koji koristi p procesora može do p puta brže obaviti izračunavanje od jednoprocesorskog sistema
    - Dodavanje više procesora nije od pomoći ako programeri nisu svesni njihovog postojanja ili ne znaju kako da ih iskoriste.
    - Sekvencijalni programi nemaju nikakve koristi od postojanja više procesora na čipu.

# Potrebe za brzim računarima

- Nauka
  - Vremenska prognoza (7-dnevna prognoza zahteva izvršenje ~ $10^{15}$ FP operacija; računaru koji može da obavi 1 GFLOPS ($10^9$ FP operacija u sec potrebno je $10^6$ sec ~10 dana) (7 dnevna prognoza za 24 sata → 56 Gflop/s)
  - modelovanje globalnih klimatskih promena (50 god. za 30 dana → 4.8 Tflop/s)
  - Astrofizika
  - Biologija, genetika, proteinski lanci...
  - Hemija: razvoj novih lekova
  - Nauka o materijalima
- Tehnika
  - Simulacije sudara – simulacija: teći stub nauke
  - Poluprovodnička tehnologija
  - Simulacija zemljotresa
  - Dinamika fluida
- Ekonomija
  - Modelovanje finansijskih transakcija
  - Elektronsko poslovanje
- Odbrana
  - Nuklearno oružje -- testiranje pomoću simulacija
  - Kriptografija

# Šta je paralelni računar

- Definicija: "Paralelni računar predstavlja skup procesnih elemenata (procesora) koji medjusobno saradjuju i komuniciraju radi bržeg rešavanja nekog problema"
- Pitanja
  - Koliko je velik taj skup?
  - Koliko su moćni procesni elementi?
  - Kako oni medjusobno saradjuju i komuniciraju?
  - Kako se podaci prenose?
  - Kako su procesni elementi medjusobno povezani?
  - Šta je na raspolaganju programeru?
  - Kakve su performanse takvih sistema?

# Na kojim nivoima se paralelizam može eksploatisati?

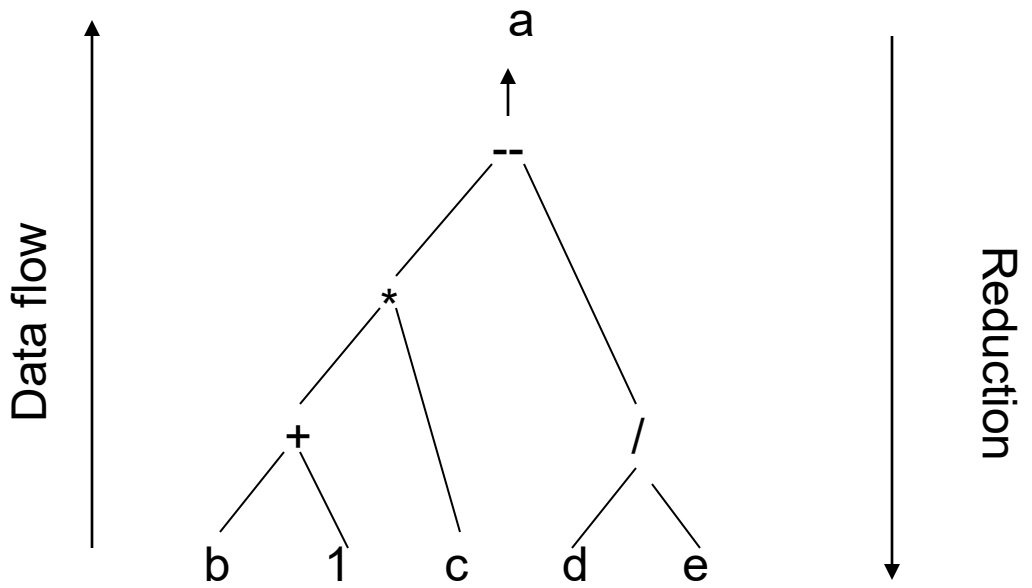
- Paralelizam na nivou bitova: 1970 do ~1985
  - 4 bits, 8 bit, 16 bit, 32, 64 bitni mikroprocesori
- Paralelizam na nivou instrukcija (ILP): ~1985 do kraja 1990ih
  - Protočnost (Pipelining) i RISC + moćni kompilatori
  - Superskalarni procesori
  - VLIW
  - Ova hardverska poboljšanja nisu skalabilna i zahtevaju sofisticirane kompajlerske tehnologije.
  - Vektorki računari takođe koriste LLP i imaju dobre performanse samo za određeni broj problema.
- Paralelizam na nivou zadataka (procesa, taskova)
  - Više procesa simultano rešava dati problem
  - Opšte prihvaćen trend kod računara opšte namene
  - Komunikacija i sinhronizacija procesa čine srž paralelnog programiranja
  - Serveri su paralelnil
  - dual, quad - core PC

# Podela PRS

- U odnosu na način upravljanja:
  - Programski upravljane (control driven Von Neumann)
    - Redosled izvršenja instrukcija odredjen je sadržajem programskog brojača.
      - Paralelne arhitekture se dobijaju povezivanjem dve ili više jednoprocesorskih mašina
      - Svaka procesorska jedinica sledi tradicionalnu sekvencu mašinskih ciklusa: pribavljanje instrukcije-izvršenje-smeštanje rezultata
  - Upravljanje definisano tokom podataka (Data flow računari)
    - Instrukcija se izvršava čim je dostupna odgovarajuća kombinacija njenih argumenata (halapljiva izračunavanja)
      - poseduju sitno zrnasti paralelizam na nivou instrukcija
  - Upravljanje po zahtevu (deamand driven, Reduction mašine)
    - Izvršavaju se samo one instrukcije čiji je rezultat potreban za druga izračunavanja.
      - Program se može posmatrati kao niz ugnježdjenih izračunavanja, a izvršenje se obavlja redukovanjem najdublje ugnježdjenih izračunavanja u saglasnosti sa semantikom odgovarajućih operatora, sve dok postoji potreba za izračunavanjem

# Data flow i Reduction prilaz

- Data flow računari koriste prilaz "odozdo-na-gore" u izračunavanju
- Reduction mašine koriste prilaz "s-vrha –na-niže"
- Primer
  - a=(b+1)*c-d/e

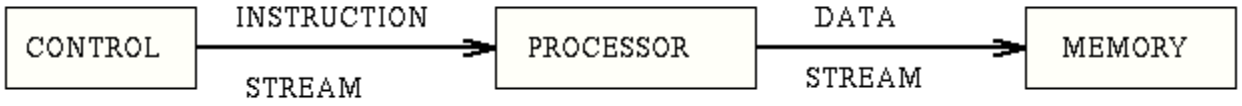


# Flinova klasifikacija Von Neumanovih računara

- Podela izvršena prema broju jednovremenih tokova instrukcija i tokova podataka
  - SISD (Single Instruction Single Data stream) – računari sa jednim tokom instrukcija i jednim tokom podataka
  - MISD (Multiple Instruction Single Data stream) – računari sa višestrukim tokom instrukcija i jednim tokom podataka
  - SIMD (Single Instruction Multiple Data stream) – računari sa jednim tokom instrukcija i višestrukim tokom podataka
  - MIMD (Multiple Instruction Multiple Data stream) – računari sa višestrukim tokom instrukcija i višestrukim tokom podataka

# Klasifikacija Von Neumanovih računara

## ✱ SISD Computers (Single Instruction Single Data stream)

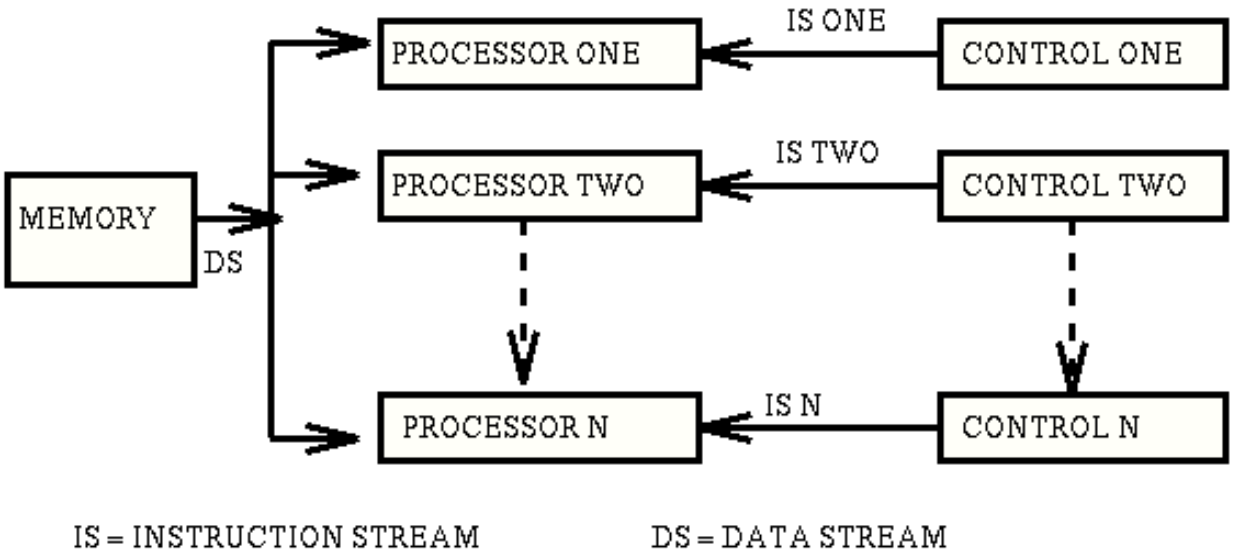



Primer:

- Brzina je ograničena internom brzinom prenosa u okviru računara
- Da bi se izračunala suma od N brojeva a1, a2, .... ,aN procesor mora pristupiti memoriji N puta uzastopno.
- Takodje, potrbno je obaviti N-1 sabiranje.
  - Zbog toga ovo izračunavanje zahteva izvršenje O(N) operacija. t.j. algoritmi za SISD računare ne sadrže ni jedan vid paralelizma jer postoji samo jedan procesor.

# MISD računari

N procesora, svaki sa sopstvenom upravljačkom jedinicom koji dele pristup zajedničkoj memoriji.

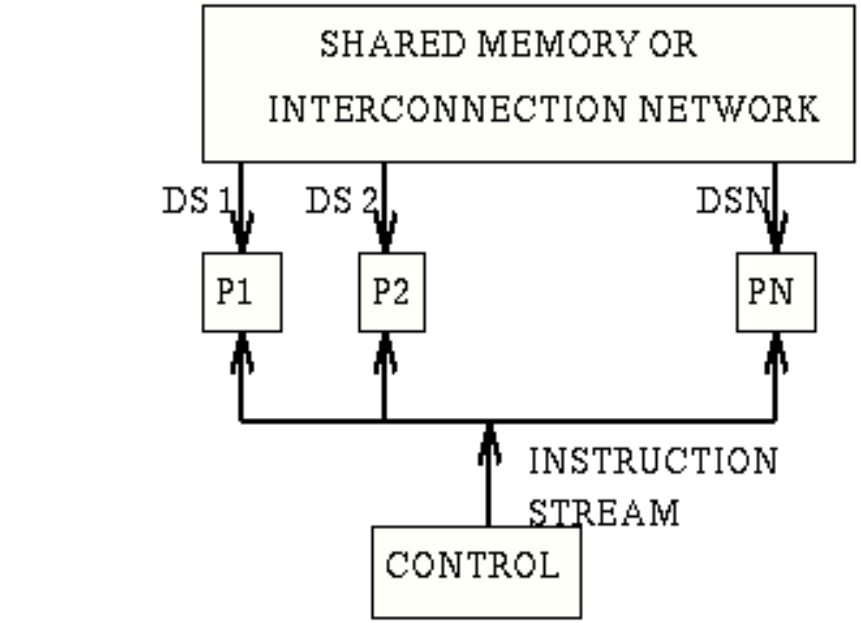


# MISD računari

- Postoji N tokova instrukcija i jedan tok (niz) podataka. Paralelizam se postiže tako što više procesora izvršava različite instrukcije u isto vreme nad istim podatkom.
- MISD mašine su od koristi kod onih izračunavanja gde isti podatak treba obraditi na više načina .
  - Primer: Testiranje na oblika.
  - Za većinu aplikacija MISD način rada nije pogodan, tako da nema komercijalno raspoloživih mašina ovog tipa.

# SIMD

Postoji N procesora čijim radom upravlja jedinstvena upravljačka jedinica (control) .Ovo je isto kao da svaki procesor sadrži identičnu kopiju programa. Postoji N tokova podataka, po jedan za svaki procesor, što znači da se različiti podaci mogu koristiti u svakom procesoru.




P = PROCESSOR
DS = DATA STREAM

# SIMD (nast.)

• Procesori u sistemu rade sinhrono: svi izvršavaju istu instrukciju nad različitim podatkom.

Procesorska polja kao npr. ICL DAP (Distributed Array Processor) spadaju u ovu kategoriju.
• SIMD mašine su naročito pogodne za probleme koji imaju regularnu strukturu, tj. ista instrukcija se može primeniti na podskup podataka.

**Primer:** Sabiranje dve matrice A + B = C.

Pretpostavimo da imamo dve matrice A i B reda 2 i 4 procesora
A11 + B11 = C11 ... A12 + B12 = C12
A21 + B21 = C21 ... A22 + B22 = C22

Ista instrukcija se izdaje svim procesorima (sabiranje dva broja) i svi procesori je izvršavaju jednovremeno.
Izračunavanje se obavlja u jednom koraku naspram 4 koraka na sekvencijalnoj mašini.

# SIMD (nast.)

Ponekad može biti potrebno da samo podskup procesora izvršava datu instrukciju (tj. samo neki podaci trebaju biti obradjeni datom instrukcijom)
Ova informacija može biti kodirana samom instrukcijom ukazujući da li je

1. procesor **aktivan** ( izvršava instrukciju)
2. procesor **pasivan** ( čeka na sledeću instrukciju)

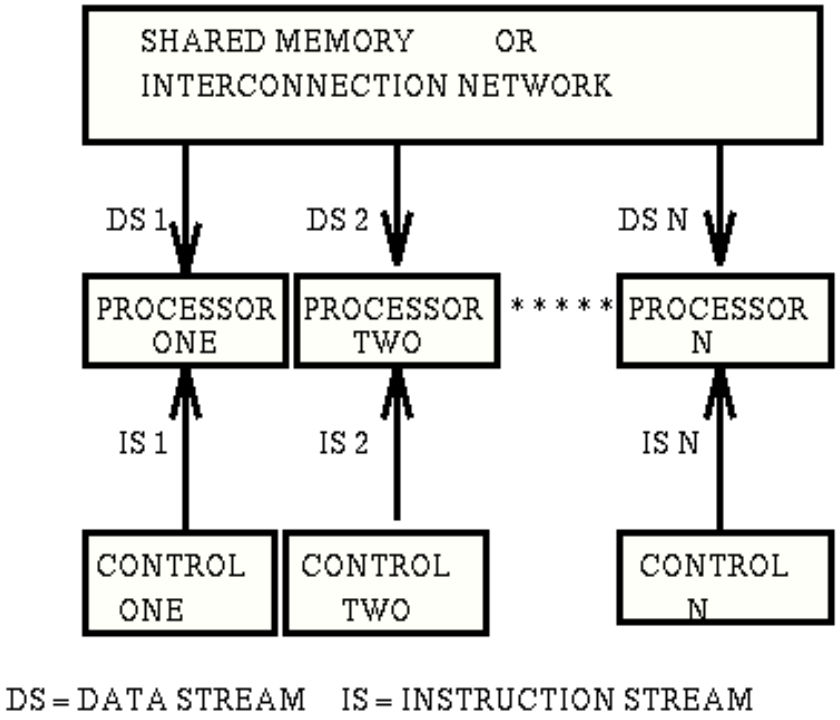
U toku izvršenja programa na SIMD (i MIMD) javlja se potreba za komunikacijom izmedju procesora da bi se razmenili podaci ili rezultati.

Komunikacija se može ostvariti korišćenjem deljive memorije i deljivih promenljivih ili slanjem poruka kroz sprežnu mrežu.

Primeri SIMD:Illiac IV, MPP, DAP, CM-2, MasPar MP-2

# MIMD računari (multiprocessors / multicomputers)

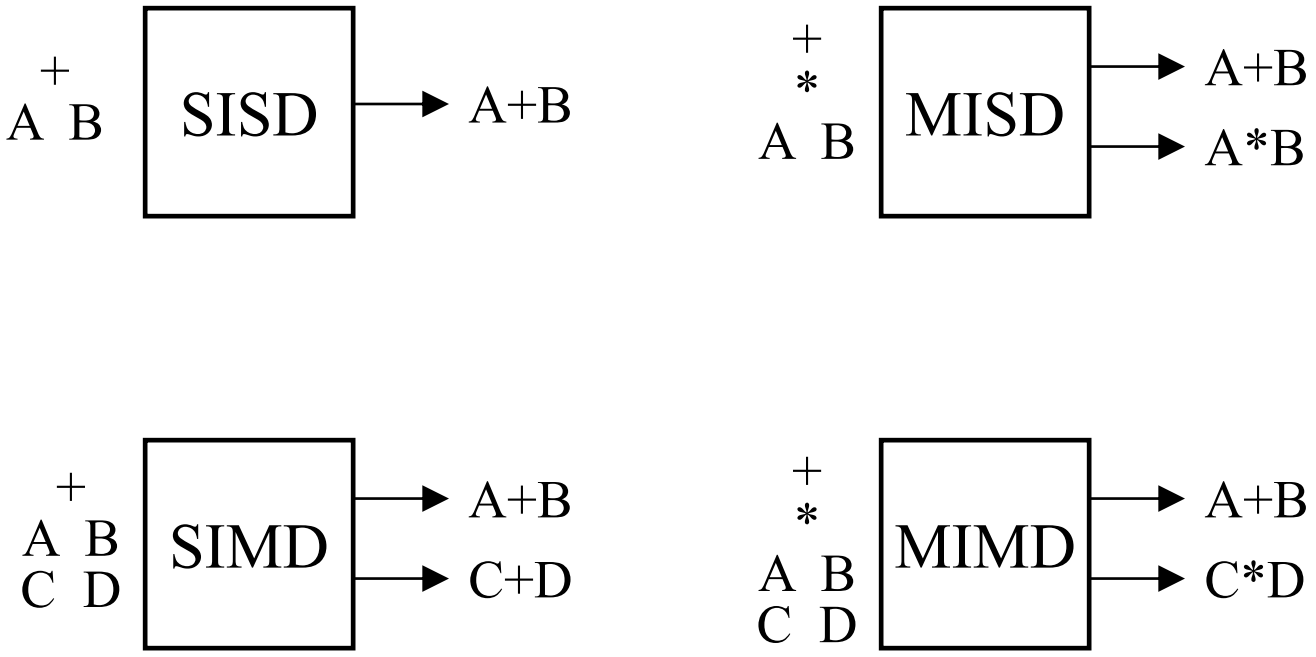
To je najopštija i najmoćnija klasa računara..
Postoji N procesora, N nizova instrukcija i N nizova podataka.



# Osobine MIMD sistema

- Procesori rade asinhrono (mogu izvršavati različite instrukcije nad različitim podacima u isto vreme)
- Komunikacija izmedju procesora se može obaviti preko deljive memorije ili slanjem poruka kroz sprežnu mrežu.
- MIMD računari sa deljivom memorijom zovu se **multiprocesori** ili **čvrsto spregnuti sistemi** (Primeri: ENCORE, MULTIMAX, SEQUENT & BALANCE.)
- MIMD računari koji komuniciraju slanjem poruka kroz sprežnu mrežu zovu se **multiračunari** ili **slabo spregnuti sistemi** (Primeri: INTEL iPSC, NCUBE/7 )

# Potencijali četiri klase računara



# Podela PS u odnosu na interprocesorsku komunikaciju

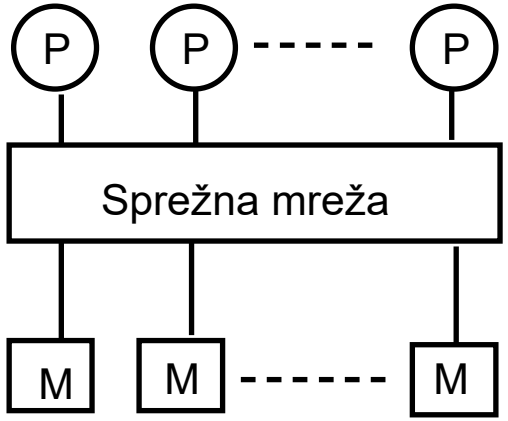
- Zajednička (deljiva) memorija
  - Deljiva memorija se sastoji od globalnog adresnog prostora. Svi procesori mogu da čitaju i vrše upis u ovaj globalni adresni prostor.
  - Memorija ne mora biti jedinstvena (može postojati više memorijskih banaka)
  - Procesori komuniciraju preko read i write operacija
- Sistemi sa razmenom poruka
  - Svaki procesor ima odvojeni (sopstveni) adresni prostor.
  - Procesor ne može pristupati adresnom prostoru drugog procesora.
  - Procesi komunicraju razmenjujući poruke

# Zajednička (delljiva) memorija

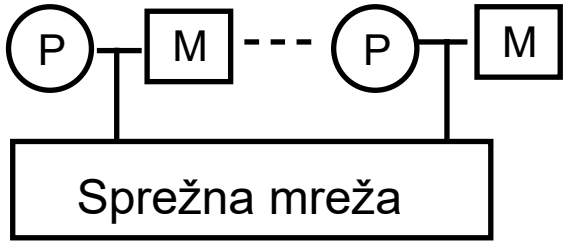
- Interprocesorska komunikacija se obavlja preko read i write operacija.
- Zajednička memorija može biti centralizovana
  - Vreme potrebno za pristup memorijskoj lokaciji je isto za sve procesore (UMA sistemi – Uniform Memory Access)
- Ili distribuirana po procesorima
  - Vreme pristpa memorijskim lokacijama može biti različito (NUMA sistemi – Non Uniform Memory Access)
    - Vreme pristupa lokalnoj memoriji je kraće od vremena pristupa memoriji drugog procesora.

# Arhitekture sa deljivom memorijom

- Ključna osobina:
  - svi procesori u sistemu mogu direktno pristupati svim memorijskim lokacijama
  - Jednostavan mehanizam interprocesorske komunikacije preko read/write operacija




UMA

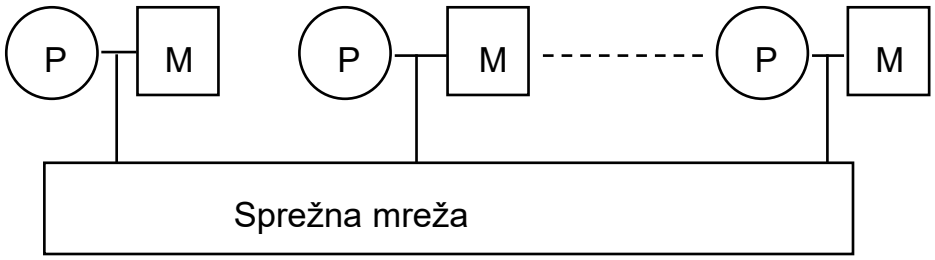



NUMA

Osnovni problem proširljivost sistema (skalabilnost)

# Arhitekture sa slanjem poruka

- Procesori mogu direktno pristupati samo svojoj lokalnoj memoriji (read/write operacije).
- Svaka komunikacija i sinhronizacija se obavlja slanjem poruka poreko send/receive operacija.

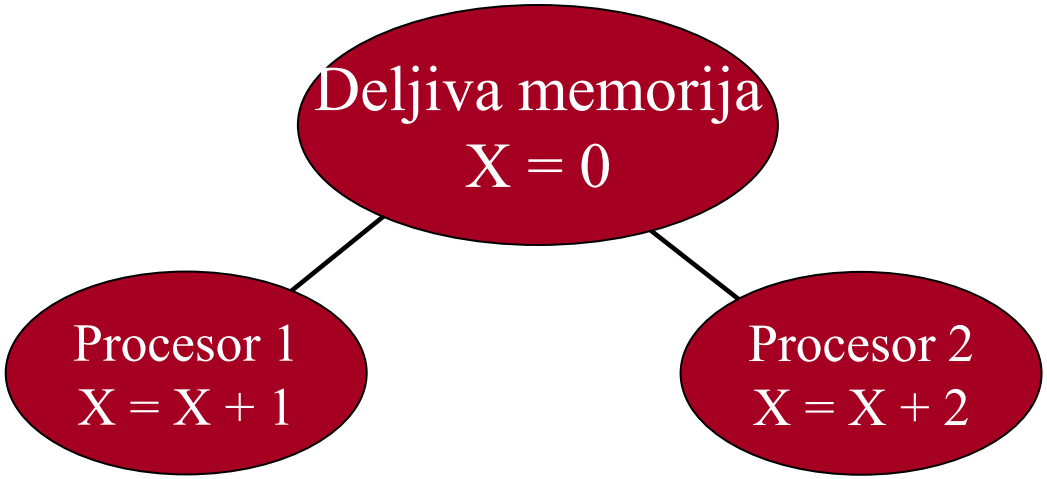



- Ključne osobine:
  - Slanje poruke podrazumeva niz aktivnosti :
    - Kreiranje zaglavlja poruke; kopiranje podataka u bafer za komunikaciju; slanje poruke; prihvatanje poruke u bafer; kopiranje poruke iz kernela u korisnički prostor**.**
      - **Mnogi od ovih koraka zahtevaju intervenciju OS.**
  - Sinhronizacija se obavlja korišćenjem nekog handshake protokola.
  - Osnovan prednost: Proširljivost sistema.

# Konflikti kod pristupa deljivoj memoriji

- Komunikacija preko deljive memorije je brža od komunikacije slanjem poruka
  - Korišćenje deljive memorije za komunikaciju može dovesti do problema kada procesori simultano pristupaju istoj memorijskoj lokaciji
- Primer:
  - Pretpostavimo da se u deljivoj memoriji nalazi promenljiva x čija je inicijalna vrednost 0.
  - Procesor P1 dodaje 1 promenljivoj x
  - Procesor P2 dodaje 2 promenljivoj x.
  - Koja je krajnja vrednost za x?

# Konflikti kod pristupa deljivoj mem. (nastavak)




Ovo je primer nedeterminisanosti u izvršenju

Moguće su sledeće vrednosti:

1. Ako P1 obavi x=x+1 pre nego što P2 pročita vrednost x izmemorije, tada je krajnja vrednost za x, 3.Slično, ako P2 obavi x+2 pre nego što P1 pročita x, krajnja vrednost za x je 3.
2. Ako P1 ili P2 pročita x iz memorije pre nego što drugi procesor obavi upis, tada krajnja vrednost x zavisi od toga koji je procesor poslednji izvršio upis

   - Ako P1 obavi poslednji upis, vrednost x je 1
   – Ako P2 obavi poslednji upis , vrednost x je 2

# Nedeterminisanost

- Nedeterminisanost je uzrokovana trkom.
  - Do trke dolazi kada dve naredbe u konkurentnim taskovima (zadacima) pristupaju istoj memorijskoj lokaciji, pri čemu je bar jedna od naredbi upis u memoriju i pri tome nema nikakvih garancija o redosledu pristupa memoriji.
- Problem nedeterminisanosti se može rešiti sinhronizacijom pristupa deljivim podacima.
  - Ako su naredbe x+1 i x+2 uzajamno isključive, tada će vrednost x uvek biti 3.
- Delovi paralelnog programa koji zahtevaju sinhronizaciju da bi se izbegla nedeterminisanost zovu se *kritične sekcije*.

# Zaključavanje i uzajamno isključivanje

U sistemima koji koriste deljivu memoriju može se koristiti mehanizam zaključavanja (lock) da bi se obezbedilo uzajamno isključivo pravo pristupa deljivim podacima.

```
Processor 1:
        LOCK (X)
                X = X + 1
        UNLOCK (X)
```

```
Processor 2:
        LOCK (X)
                X = X + 2
        UNLOCK (X)
```

# Apstraktni model računara sa deljivom memorijom (PRAM – Parallel Random Access Machine) – Idealni paralelni sistem

- Apstraktni model pokazao se veoma korisnim kod procene performansi paralelnih algoritama nezavisno od realne mašine.
- PRAM razlikuje 4 modela računara sa deljivom memorijom u zavisnosti od toga da li dva ili više procesora mogu simultano pristupati deljivoj memoriji radi čitanja ili upisa:
  - Exclusive Read, Exclusive Write (EREW)
    - Pristup memorijskoj lokaciji je ekskluzivan, tj. procesori ne mogu jednovremeno pristupati istoj mem. lokaciji ni radi čitanja ni radi upisa.
  - Concurrent Read, Exclusive Write (CREW)}
    - Više procesora može čitati istovremeno sa iste mem. lokacije, ali je upis ekskluzivan.
  - Exclusive Read, Concurrent Write (ERCW)}
    - Više procesora može jednovremeno vršiti upis u istu mem. lokaciju, a čitanje je ekskluzivno.
  - Concurrent Read, Concurrent Write (CRCW)}
    - Dozvoljen je jednovremeni pristup i radi čitanja i radi upisa.

# Komentari o Shared memory računarima (1)

- Simultano čitanje iste memorijske lokacije ne dovodi do problema.
  - Svaki procesor napravi kopiju sadržaja mem. lokacije i zapamti je u svom registru.
- Problem se javlja kod konkurentnog upisa.
  - Ako više procesora pokušava da izvrši upis u istu mem. lokaciju kome dozvoliti upis?
- Postoji nekoliko načina da se deterministički odredi sadržaj mem. lokacije nakon konkurentnog upisa:
  - Dodeliti prioritete procesorima i zapamtiti vrednost procesora sa najvišim prioritetom..
  - Svim procesorima je dozvoljeno da izvrše upis pod uslovom da su sve vrednosti jednake.
  - Upisuje se  max, min, sum, ili srednja vrednost podataka (ako su u pitanju numerički podaci)

# Primer

Da bi pokazali kako se 4 podklase SM mašina ponašaju, razmotrimo sledeći primer.

**Problem:**

Zadata je lista od m elemenata S = {$L_1$, $L_2$, …,$L_m$}. Potrebno je naći indeks (redni broj) zadatog elementa x. Element x se može pojaviti više puta u S i bilo koja vrednost rednog broja je prihvatljiva. Na raspolaganju je N procesora, $1 < N \leq m$.

# Algoritam

**procedure** SM_search (S, x, k)

**STEP 1**: **for** i=1 **to** N **do in parallel**

read x

**end for**

**STEP 2**: **for** i=1 **to** N **do in parallel**

$S_i = \{L_{((i-1)m/N+1)}, \ldots, L_{(im/N)}\}$

obaviti sekvencijalno pretraživanje u podlisti $S_i$

(vratiti $K_i$= -1 ako x nije u listi, inače $K_i$= index)

**end for**

**STEP 3**: **for** i=1 **to** N **do in parallel**

**if** $K_i > 0$ **then** $k=K_i$ **end if**

**end for**

**end** procedure

# Vremenska kompleksnost za EREW

Ako sekvencijalno pretraživanje podliste traje O(m/N) vremenskih jedinica, kolika je vremenska kompleksnost svake od 4 podklase shared memory computer?

- EREW

  Step 1 traje O(N) (N čitanja, jedno u jednom trenutku).

  Step 2 traje O(m/N) vremena.

  Step 3 traje O(N) vremena.

  Ukupno vreme O(N)+O(m/N).

# ERCW

- ERCW

  Step 1 traje O(N) vremena.

  Step 2 traje  O(m/N) vremena.

  Step 3 konstantno vreme.

  Ukupno: O(N)+O(m/N).

# CREW

- CREW

  Step 1 konstantno vreme.

  Step 2 traje O(m/N) vremena.

  Step 3 traje O(N) vremena

  Ukupno: O(N)+O(m/N).

# CRCW

- CRCW

  Step 1 konstantno vreme.

  Step 2 traje O(m/N) vremena.

  Step 3 konstantno vreme.

  Ukupno: O(m/N).

# Ograničenja Shared memory

- SIMD mašine se obično sastoje od nekoliko 1000 veoma jednostavnih procesora.
  - Shared memory SIMD mašine nisu realne zbog cene i teškoća oko realizacije pristupa deljivoj memoriji za veliki broj procesora. Zbog toga ne postoje komercijalno raspoložive SIMD mašine sa deljivom memorijom.
- MIMD mašine koriste moćnije procesore, a MIMD sa deljivom memorijom postoje u sistemima sa relativno malim brojem procesora (do 100)

# Šta je programeru na raspolaganju?

- Sekvencijalni programski jezici – od paralalizirajućeg kompajlera se traži da konvertuje program u paralelni i generiše izvršni kod
  - prilaz koji se dans retko koristi
- Sekvencijalni programski jezici dopunjeni konstrukcijama za deklaraciju deljivih promenljivih i paralelizma
  - Npr UPC (Unified Parallel C) – zahteva UPC kompajler, ACTUS.
- Sekvencijalni PJ sa preprocesorskim kompajlerskim direktivama za deklaraciju deljivih promenljivih i paralelizma
  - Npr. OpenMP – industrijski standard - potreban OpenMP kompajler
- Niti (Threads )- programer vrši dekompoziciju programa na paralelne sekvence (niti) pri čemu svaka nit može da pristupa globalnim promenljivim
  - Npr. Pthreads
- Paralelni programski jezici - kompajler kreira izvršni kod za svaki procesor
  - Npr. Ada
- Biblioteka funkcija za interprocesorsku komunikaciju koje se mogu pozivati iz različitih programskih jezika (C, C++, Fortan, ...)
  - MPI (Message-Passing Interface) standard definisan 1990ih.
  - PVM (Parallel Virtual Machine) – razvijen krajem 1980-ih

# Performanse paralelnih sistema

Ubrzanje, Efikasnost

# Ubrzanje sistema

- Kod arhitektura kod kojih je uveden bilo koji vid poboljšanja, može se dati ocena o dobijenom poboljšanju sa stanovišta performansi korišćenjem mere UBRZANJE

  - $S = \frac{\text{Vreme izvršenja programa na arhitetkuri bez poboljšanja}}{\text{Vreme izvršenja programa na arh. sa izvedenim poboljšanjem}}$

  - Za paralelni sistem sa n procesora

    - $S(n) = \frac{T(1)}{T(n)}$

      - T(1) vreme izvršenja programa na jednoprocesorskom sistemu
      - T(n) vreme izvršenja programa na n-procesorskom sistemu

# Ubrzanje (nast.)

$$T(1) = T_s + T_p$$

$$T(n) = T_s + \frac{T_p}{n} + T_o(n)$$

- Amdahl-ov zakon (krajem 60-ih god. 20.v)
  - Postavlja gornju granicu ubrzanja koje se može postići paralelnom obradom

$$T(1) = T_s + T_p, \quad T_s = \alpha T(1), \quad 0 \le \alpha \le 1$$

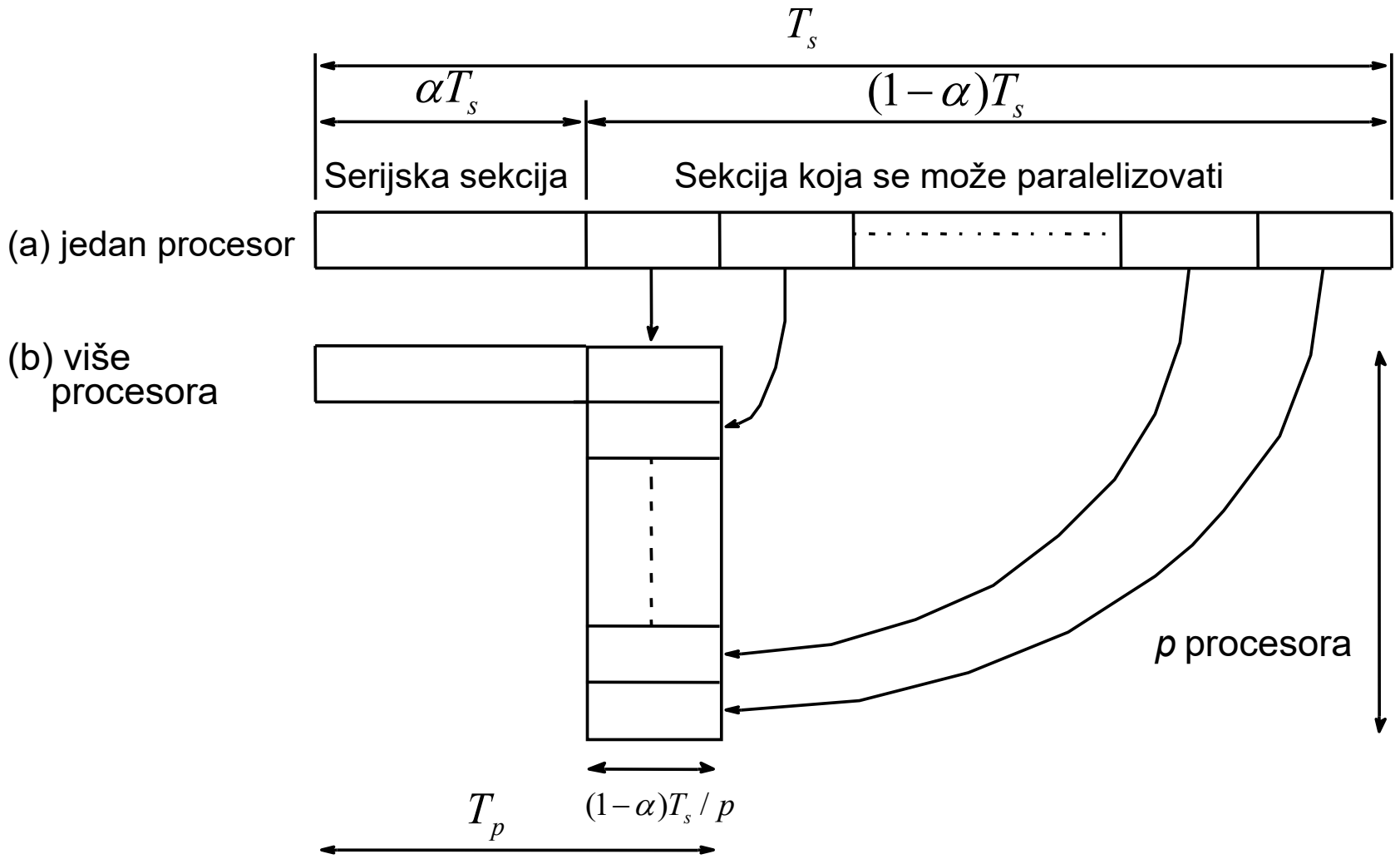
$$T(n) = T_s + \frac{T_p}{n} = \alpha T(1) + \frac{(1-\alpha)T(1)}{n}$$

$$S(n) = \frac{T(1)}{T(n)} = \frac{n}{1+(n-1)\alpha}$$

$$\lim_{n\to\infty} S(n) = \frac{1}{\alpha}$$

Npr .$\alpha$=1/20 (5%), S(n)=20

# Amdalov zakon



# Da li je Amdahl bio u pravu?

- $\alpha$ nije konstantno
- $\alpha = \alpha(m)$, m – obim problema
- Kod najvećeg broja naučno-tehničkih aplikacija

  $\lim_{m \to \infty} \alpha(m) = 0$, tj. sa porastom obima problema $\alpha$ (Amdahlova frakcija) teži 0.

  - Algoritmi sa ovom osobinom zovu se efektivni paralelni algoritmi
  - Efektivni algoritmi imaju linerano ubrzanje

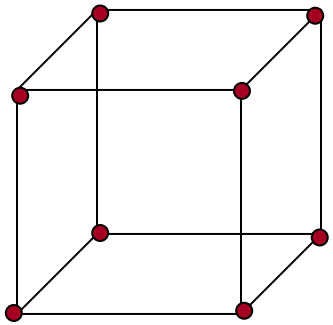
$$\lim_{m \to \infty} S(n) = \lim_{m \to \infty} \frac{n}{1 + (n-1)\alpha(m)} = n$$

# Primer efektivnog paralelnog algoritma

## ✱ Proizvod matrice i vektora

$$\vec{Y} = A\vec{x}, \quad A = (a_{ij})_{m\times m} \quad \vec{x} = (x_j)_{m\times 1}$$

- Neka se program izvršava na $n=2^r$ hiperkubu
  - max. rastojanje izmedju čvorova $\log_2 n = r$
  - svaki čvor je direktno povezan sa r najbližih suseda



hiperkub za n=8

# proizvod matrice i vektora (nast.)

- Neka je vektor X zapamćen u lokalnoj memoriji prvog procesora
- Neka se u lok. mem. svakog procesora nalazi $k \leq [m/n]+1$ vrsta matrice A

* Efektivni paralelni algoritam

1. distribuirati vektor x svim ostalim procesorima
2. svi procesori jednovremeno računaju komponente vektora y

$$y_i = \sum_{j=1}^{m} a_{ij} x_j, \quad i = 1,\ldots,k$$

3. sve komponente vektora y se predaju prvom procesoru

- Pokazaćemo da je ovaj algoritam efektivan,
  - tj. $\lim_{m\to\infty} \alpha(m)=0$

# proizvod matrice i vektora (nast.)

- Uticaj na faktor $\alpha(m)$ imaju aktivnosti koje se odnose na:
  - predaju vektora x
  - debalans u opterećenju, ako m/n nije ceo broj
  - predaja komponenti vektora y jednom (prvom) procesoru
- Trajanje koraka
  1. predaja vektora x ostalim procesorima $O(m*\log_2 n)$ koraka
  2. Vreme potrebno za izračunavanje k elemenata vektora y $O(km)=O(m^2/n)+O(m)$
  3. Predaja komponenti vektora y prvom procesoru $O(m*\log_2 n)$

$$T_s = \alpha \cdot T(1), \Rightarrow \alpha = \frac{T_s}{T(1)}$$

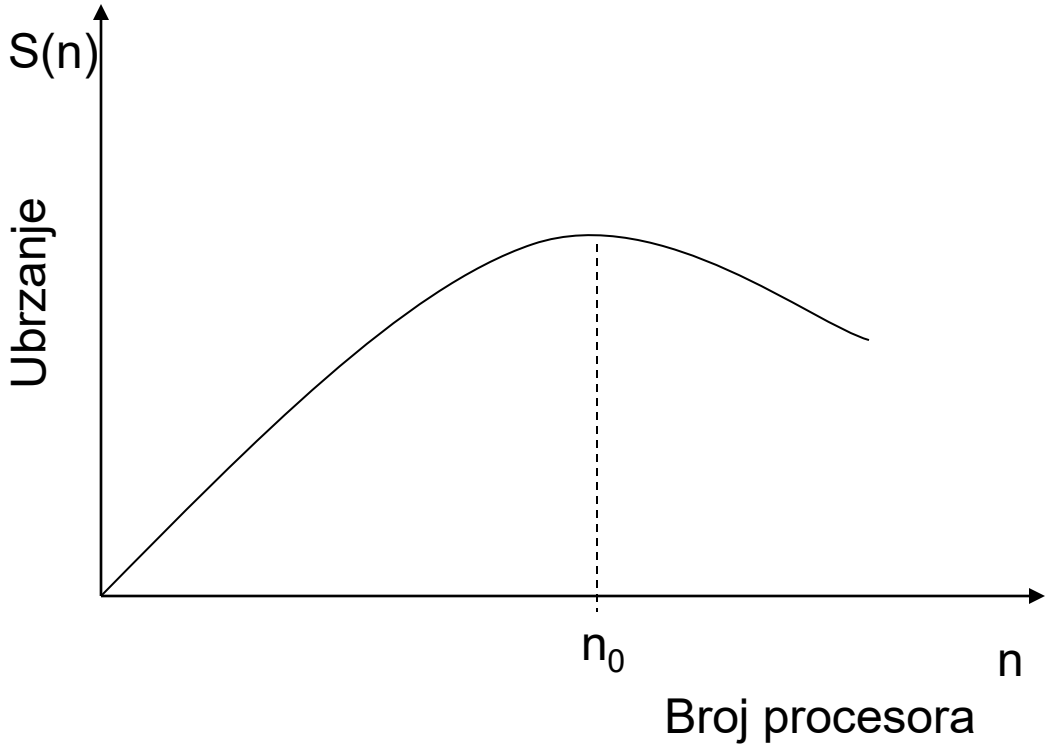
$$T(1) = O(m^2)$$

$$T_s = O(m \log_2 n) + O(m)$$

$$\alpha = \frac{T_s}{T(1)} = \frac{O(m \log_2 n) + O(m)}{O(m^2)} = O\left(\frac{1 + \log_2 n}{m}\right)$$

$$\lim_{m \to \infty} \alpha(m) = 0$$

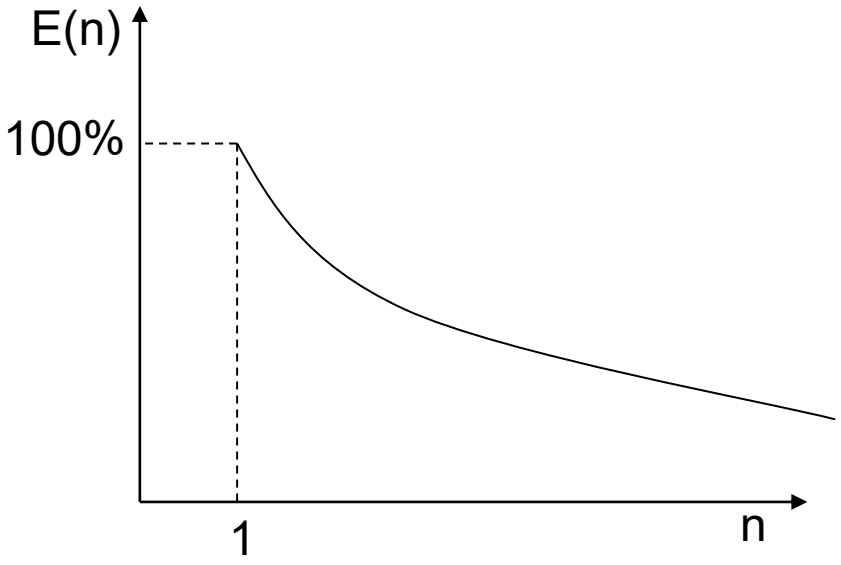
# Šta je realnost

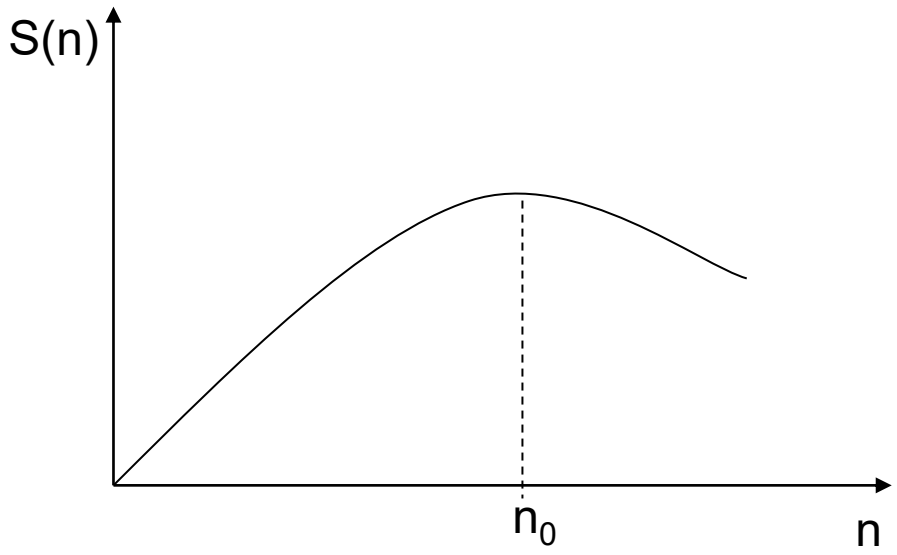


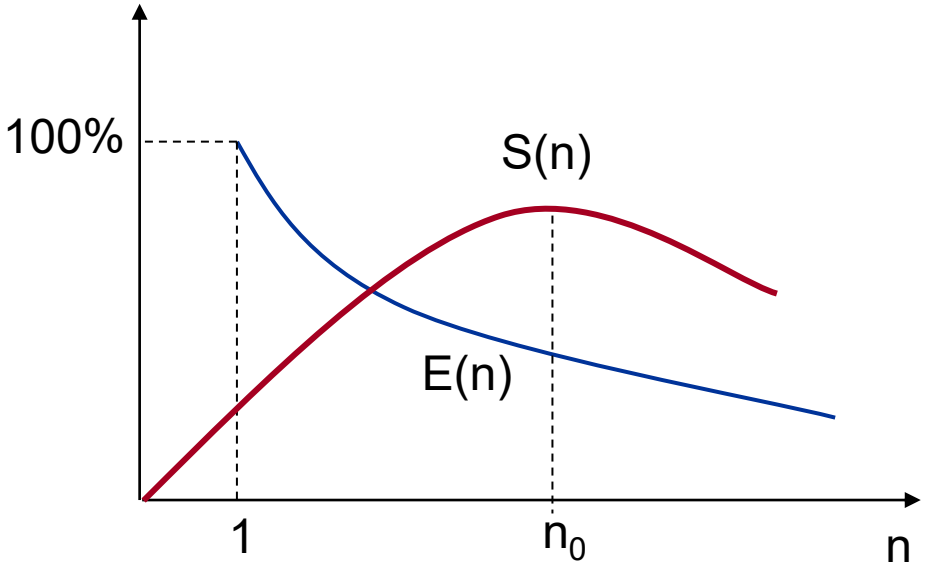
# Efikasnost

- Definiše se kao odnos ubrzanja i broja procesora
  - $E(n)=S(n)/n, \quad 0 \leq E \leq 1$
  - Daje ocenu o srednjoj iskorišćenosti procesora u sistemu kada su svi procesori angažovani na rešavanju istog problema
    - ako se zanemare U/I aktivnosti, efikasnost jednoprocesorskig sistema je 1 (100%)
- Ubrzanje i efikasnost ne treba posmatrati odvojeno
  - sa poarstom broja procesora ubrzanje raste (do odredjene granice)
  - sa porastom broja procesora efikasnost opada
  - S(n)*E(n) je najbolja ocena

# Efikasnost i ubrzanje



# Primer: Sabiranje brojeva

$$A = \sum_{i=1}^{16} a_i$$

Posmatramo vreme izračunavanja na sistemu sa:

1. P=1
2. P=2
3. P=3
4. P=4
5. P=8 procesora

# Sistem sa jednim procesorom

| | P1 | T |
|---|---|---|
| 1. korak | A=a1+a2+…+a16 | 15 |

T=15

# Sistem sa dva procesora

| | P1 | P2 | T |
|---|---|---|---|
| 1. korak | B1=a1+a2+…+a8 | B2=a9+a10+…+a16 | 7 |
| 2. korak | A=B1+B2 | | 1 |

T=8

# Sistem sa tri procesora

| | P1 | P2 | P3 | T |
|---|---|---|---|---|
| 1. korak | B1=a1+…+a5 | B2=a6+…+a10 | B3=a11+…+a15 | 4 |
| 2. korak | C1=B1+B2 | C2=B3+a16 | | 1 |
| 3. korak | A=C1+C2 | | | 1 |

T=6

# Sistem sa četiri procesora

| | P1 | P2 | P3 | P4 | T |
|---|---|---|---|---|---|
| 1. korak | B1=a1+...+a4 | B2=a5+...+a8 | B3=a9+...+a12 | B4=a13+...+a16 | 3 |
| 2. korak | C1=B1+B2 | C2=B3+B4 | | | 1 |
| 3. korak | A=C1+C2 | | | | 1 |

T=5

# Sistem sa osam procesora

| | P1 | P2 | ... | P8 | T |
|---|---|---|---|---|---|
| 1. korak | B1=a1+a2 | B2=a3+a4 | ... | B8=a15+a16 | 1 |
| 2. korak | C1=B1+B2 | C2=B3+B4 | ... | | 1 |
| 3. korak | D1=C1+C2 | D2=C3+C4 | | | 1 |
| 4. korak | A=D1+D2 | | ... | | 1 |

T=4

- Hajdnov efekat

# Poređenje performansi

| P | Sp | Ep | SpEp |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 1.875 | 0.94 | 1.76 |
| 3 | 2.5 | 0.83 | 2.075 |
| 4 | 3 | 0.75 | 2.25 |
| 8 | 3.75 | 0.47 | 1.76 |