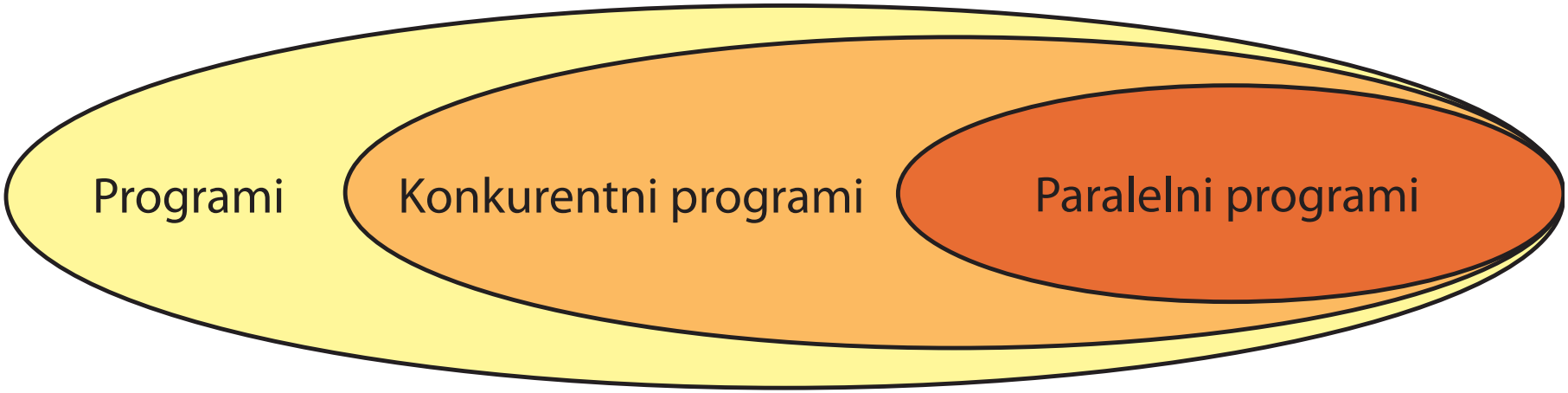
Uvod u paralelne i distribuirane sisteme

Konkurentna, paralelna i distribuirana obrada

- Odnos svih programa, konkurentnih programa i paralelnih programa:



Konkurentna, paralelna i distribuirana obrada

- **Konkurentno** – dešava se **tokom istog vremenskog intervala**
- **Paralelno** – dešava se **u isto vreme**. Paralelna obrada po definiciji zahteva veći broj procesora ili jezgara. U ovakvoj situaciji, više od jednog konkurentnog procesa može se **istovremeno** izvršavati. Kao i kod konkurentne obrade, ne mora biti komunikacije niti koordinacije između procesa
- **Distribuirano** – **više programa** izvršava se **konkurentno** i **međusobno komunicira** kako bi zajedno izvršili neko izračunavanje. Suština distribuirane obrade je u tome da se **rezultujuće izračunavanje distribuira između više procesa** koji **međusobno komuniciraju**

Paralelni i distribuirani algoritmi

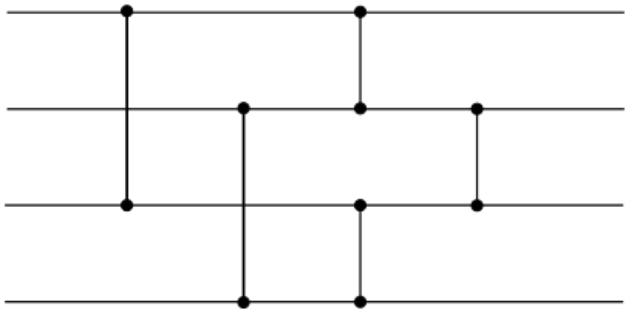
- Model **paralelnih algoritama** sa **deljenom memorijom**
  - Model je **PRAM** (engl. *parallel random access machine*)
- Model **paralelnih algoritama** sa **slanjem poruka**
  - Modeli su **Bulova logička kola** (engl. *Boolean circuits*) i **mreže za sortiranje** (engl. *sorting networks*)
- Model **distribuiranih algoritama** sa **slanjem poruka**
  - Model je **graf** u kome je **svaki čvor konačni automat** (engl. *finite-state machines*)

Paralelni i distribuirani algoritmi

- **Paralelni algoritmi sa deljenom memorijom**
  - Svi procesori imaju pristup deljenoj memoriji. Projektant algoritma bira program koji izvršava svaki od procesora
  - Teoretski model: **paralelna mašina sa slučajnim pristupom** (engl. *parallel random access machine* – **PRAM**)
  - PRAM je **apstraktna mašina sa deljenom memorijom**, analogna RAM, **zanemaruje sinhronizaciju i komunikaciju**, cena algoritma se iskazuje kroz dve mere: $O(vreme)$ i $O(vreme \times brojProcesora)$
  - Programi u deljenoj memoriji mogu da se **prošire na distribuirane** sisteme ako **operativni sistem enkapsulira komunikaciju** između čvorova i **virtuelno objedinjuje memoriju** pojedinačnih sistema
  - Model sa asinhronom deljenom memorijom je bliži realnim sistemima pošto uzima u obzir mašinske instrukcije kao što su uporedi i zameni (engl. *compare-and-swap* – CAS)

Paralelni i distribuirani algoritmi

- **Paralelni algoritmi sa slanjem poruka**
  - Projektant algoritma bira strukturu mreže, kao i programe koje izvršava svaki od računara
  - Koriste se modeli kao što su **Bulova logička kola** (engl. *Boolean circuits*) i **mreže za sortiranje** (engl. *sorting networks*). Bulova kola mogu da se posmatraju kao računarska mreža: svaki gejt odgovara računaru koji izvršava jako jednostavan program. Mreže za sortiranje takođe mogu da se posmatraju kao računarske mreže: svaki komparator je računar



Izvor: https://en.wikipedia.org/wiki/Sorting_network

Paralelni i distribuirani algoritmi

- **Distribuirani algoritmi sa slanjem poruka**
  - Projektant algoritma bira samo program. Svi računari izvršavaju isti program. Sistem mora da radi ispravno nezavisno od strukture mreže
  - Model **grafa** u kome se svaki od **čvorova** predstavlja **konačni automat**, **potezi** odgovaraju **komunikacionim kanalima**
  - Algoritmi specifični za distribuirano okruženje:
    - **Slika sistema** (engl. *snapshot*) – Chandy-Lamport algoritam beleži konzistentno globalno stanje svih procesa i poruka u asinhronom distribuiranom sistemu
    - **Konsenzus** (engl. *consensus*) – Bitcoin proof-of-work algoritam
    - **Samo-stabilizujući** (engl. *self-stabilizing*) – Dijksta Token Ring algoritam za međusobno isključivanje (Distribuirani sistem je samo-stabilizujući ako uvek završava u ispravnom stanju, nezavisno od stanja u kome je inicijalizovan i to ispravno stanje se dostiže u konačnom broju koraka)
  - Asinhrona priroda distribuiranih sistema:
    - **Sinhronizatori** se koriste kako bi se sinhroni algoritmi izvršavali u asinhronim sistemima
    - **Logički satovi** pružaju kauzalno uređenje događaja (šta se desilo pre čega)
    - **Algoritmi za sinhronizaciju satova** pružaju globalno konzistentne fizičke vremenske otiske (engl. *time stamps*)

Složenost izračunavanja algoritama

- Kod **paralelnih algoritama**, još jedan resurs koji se posmatra prilikom analize složenosti, pored vremena i prostora, je i **broj računara**
- Ako problem odlučivanja može da se reši u **polilogaritamskom vremenu** primenom **polinomnog broja procesora** kaže se da je problem u **klasi NC** ("*Nick's Class*" – Stephen Cook dao ime po Niku Pipengeru). U teoriji kompleksnosti, klasa **NC** je **skup problema odlučivanja** koji su odlučivi u **polilogaritamskom vremenu** na **paralelnom računaru** sa **polinomnim brojem procesora**. Problem je u klasi **NC** ako postoje konstante c i k takve da problem može da se reši u vremenu $\mathrm{O}(\log^c n)$ primenom $O(n^k)$ paralelnih procesora
- Klasa NC može se jednako dobro definisati primenom formalizama PRAM-a ili Bulovih kola
- Nerešen problem: $\mathrm{NC} \stackrel{?}{=} \mathrm{P}$

Složenost izračunavanja algoritama

- Prilikom analize **distribuiranih algoritama**, **više pažnje** se obično posvećuje **operacijama komunikacije** nego izračunavanja
- Možda najprostiji model distribuiranog izračunavanja je **sinhroni sistem** u kome **svi čvorovi rade sa istim korakom** (engl. *lockstep*)
- Ovaj model je poznat kao **LOCAL**
  - Tokom svake runde komunikacije, svi čvorovi paralelno:
    1) Prime poslednju poruku od svojih suseda
    2) Izvrše odgovarajuće lokalno izračunavanje
    3) Pošalju novu poruku svojim susedima
  - Kod ovakvih sistema, **centralna mera složenosti** je **broj sinhronih komunikacionih rundi neophodnih za kompletiranje zadatka**

Paralelni sistemi

Hijerarhija apstrakcija

Nivo 5 — Nivo programskih jezika

Prevođenje (kompajler)

Nivo 4 — Nivo asemblerskog jezika

Prevođenje (asembler)

Nivo 3 — Nivo mašine operativnog sistema

Delimična interpretacija (operativni sistem)

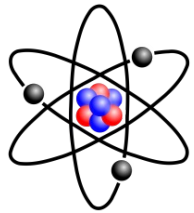
Nivo 2 — Nivo arhitekture skupa instrukcija (ISA)

Interpretacija (mikroprogram)

Nivo 1 — Nivo mikroarhitekture

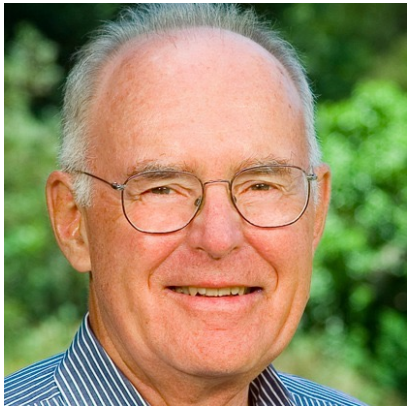
Hardver

Nivo 0 — Nivo digitalne logike



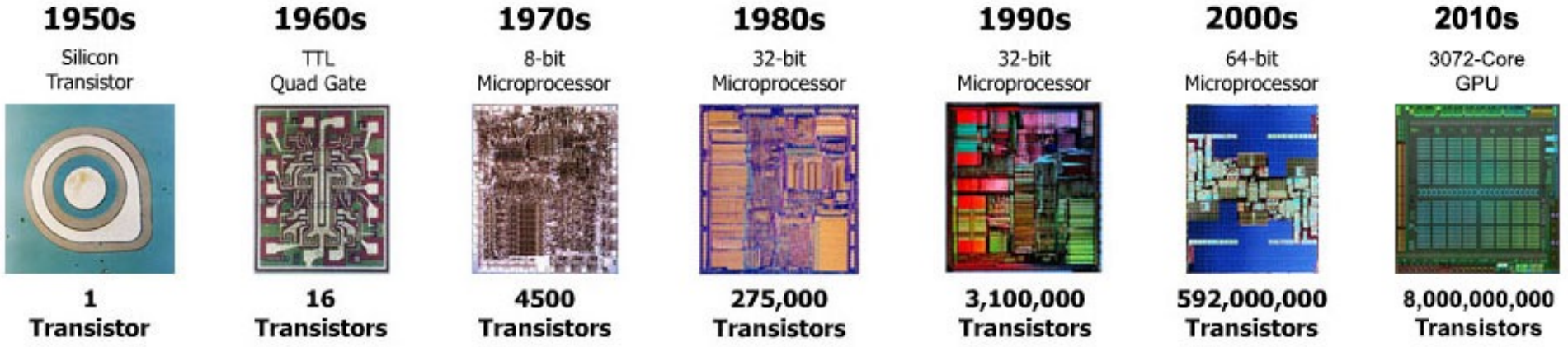
Porast performansi računara

- **Murov zakon** je zapažanje da se broj tranzistora u integrisanim kolima duplira približno svake 2 godine.



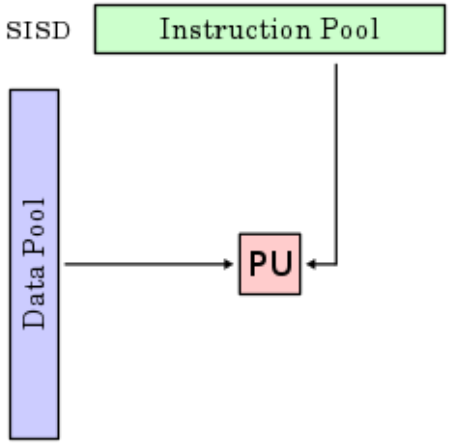
Gordon Moore (1929–)

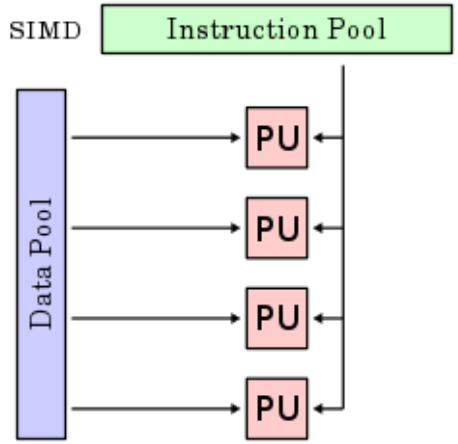
- **Evolucija arhitekture elektronskih računara (1946-danas)**



Flinova taksonomija

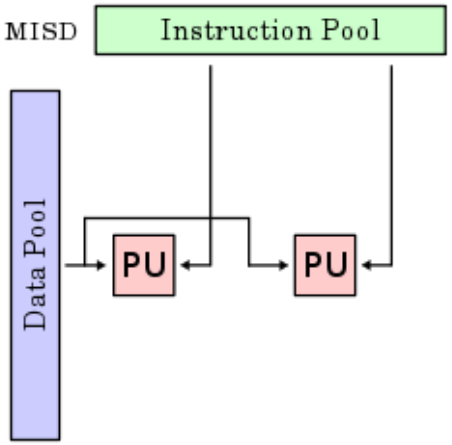
Single Instruction, Single Data (SISD)

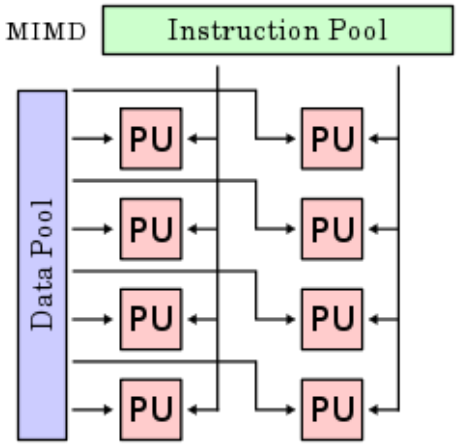




Single Instruction, Multiple Data (SIMD)

Multiple Instruction, Single Data (MISD)

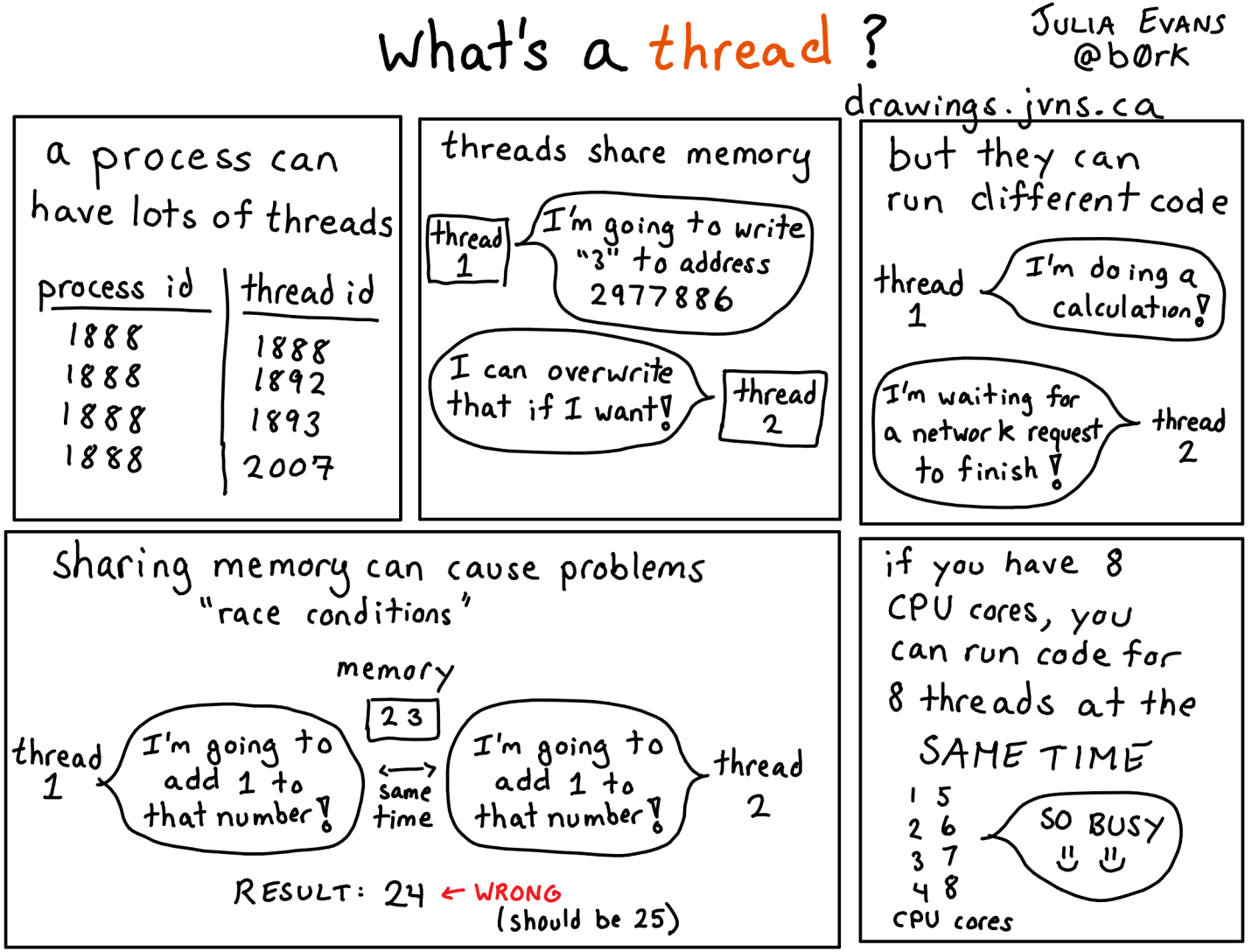




Multiple Instruction, Multiple Data (MIMD)

Izvor: https://en.wikipedia.org/wiki/Flynn%27s_taxonomy

Procesi i niti



Izvor: https://drawings.jvns.ca/drawings/threads.svg

Amdalov zakon

- Dometi primene paralelizma:
  - **Amdalov zakon** (G. M. Amdahl, 1967)
  - **Odnos serijskog i paralelnog dela algoritma**
- **Paralelna obrada ima smisla** ako je
  - **kratak sekvencijalni deo**
  - **visok stepen paralelizma**
- Potencijalno ubrzanje algoritma primenom paralelizma:

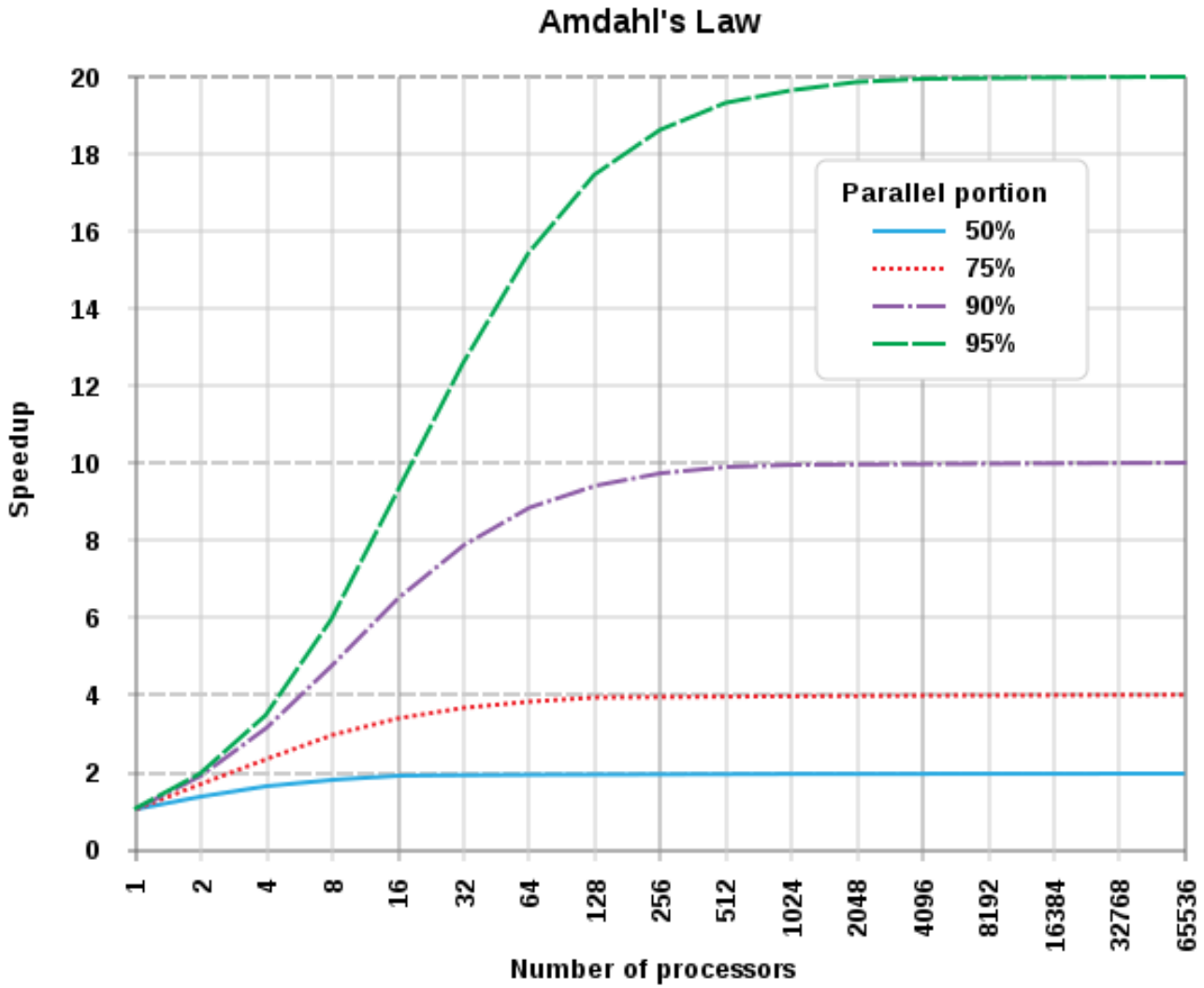
Gene Amdahl
(1922–2015)

$$S_{\text{latency}}(s) = \frac{1}{1 - p + \frac{p}{s}},$$

gde je

- S_{latency} je potencijalno ubrzanje latencije izvršavanja celog zadatka
- s je ubrzanje latencije izvršavanja dela zadatka koji može da se paralelizuje
- p je procenat vremena izvršavanja celog zadatka koji se odnosi na deo koji može da se paralelizuje *pre paralelizacije*

Amdalov zakon



Izvor: https://en.wikipedia.org/wiki/Parallel_computing

Gustafson-Barsisov zakon

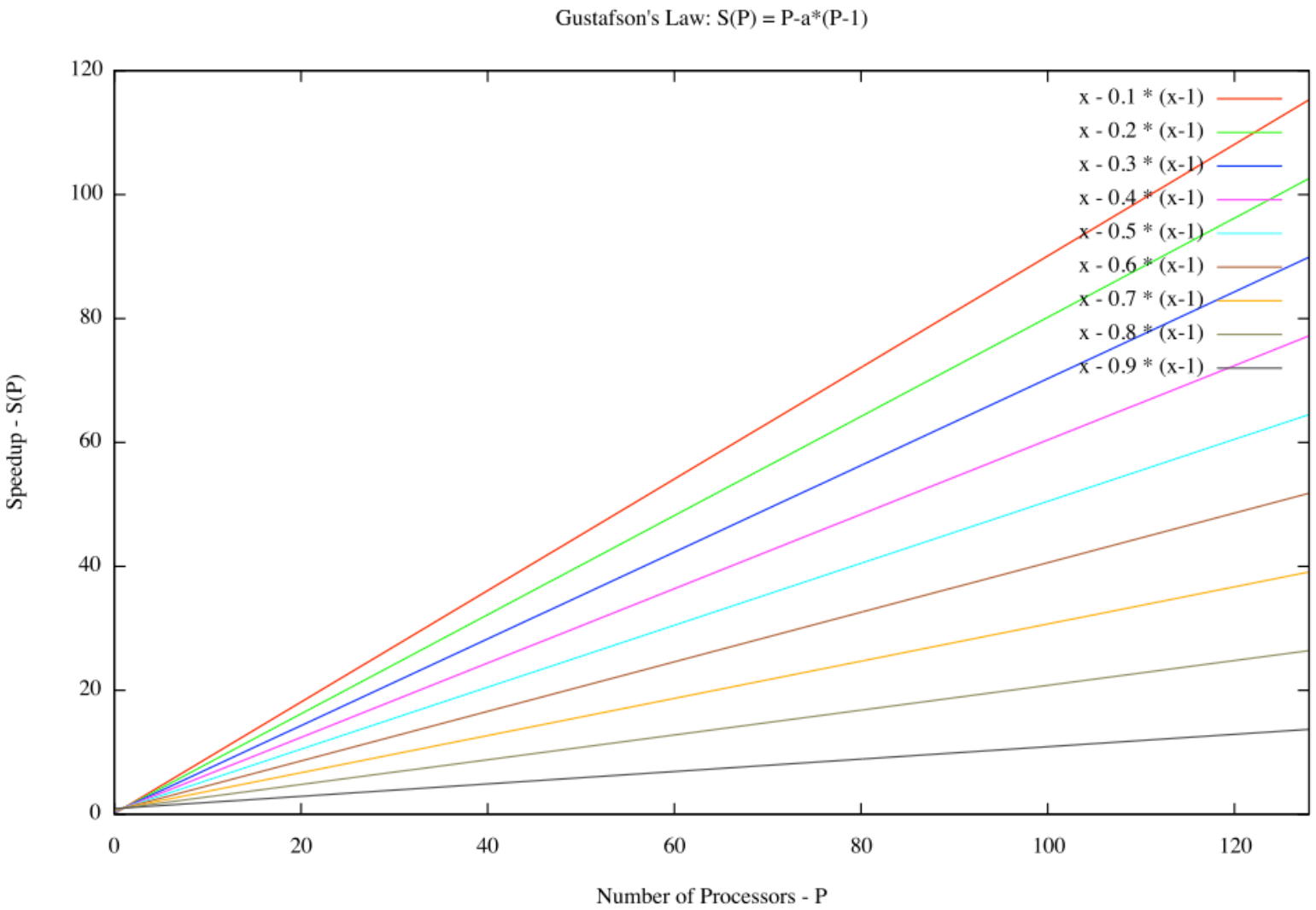
- Gustafson i Barsis, za razliku od Amdala, posmatraju **porast moći obrade** koji dovodi do toga da se **više podataka kompletnije analizira**, a **vreme obrade je fiksirano** (http://www.johngustafson.net/pubs/pub13/amdahl.htm)
- Ako su dostupni brži ili veći resursi, onda se i veće instance problema mogu rešiti za isto vreme
- Ograničenja postavljen usled nužno sekvencijalnog dela programa, mogu se delom prevazići povećanjem ukupne količine izračunavanja
- **Gustafson-Barsisov zakon**:

$$S_{\text{latency}}(s) = 1 - p + sp,$$

gde je

- S_{latency} teoretsko ubrzanje latencije izvršavanja celog zadatka
- s je ubrzanje latencije izvršavanja onog dela zadatka koji **može da iskoristi** poboljšanje u resursima dostupnim u sistemu
- p je procenat izvršnog opterećenja (engl. *execution workload*) u odnosu na celi zadatak koji se odnosi na deo kojim može da iskoristi poboljšanje u resursima sistema **pre poboljšanja**

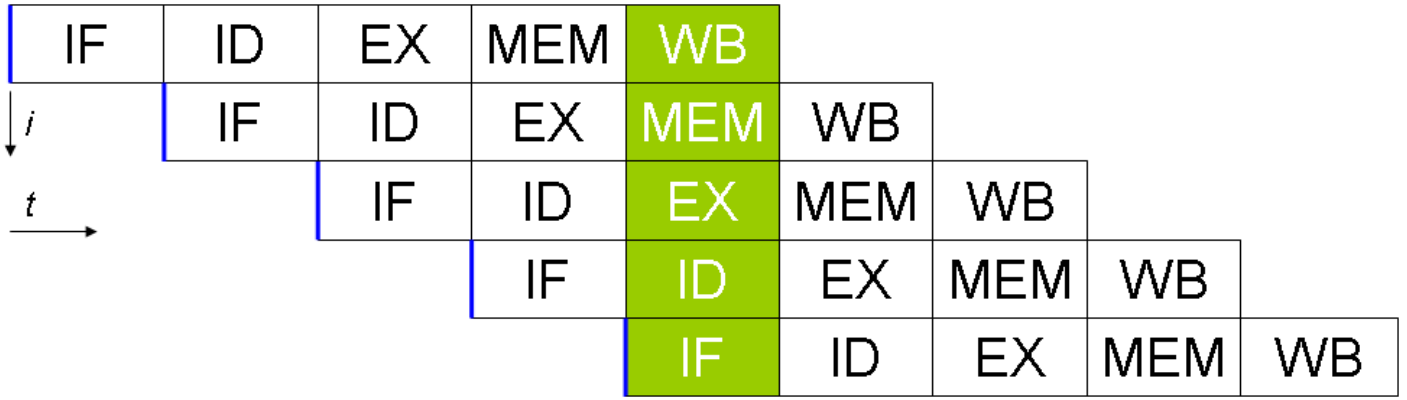
Gustafson-Barsisov zakon



Izvor: https://en.wikipedia.org/wiki/Gustafson%27s_law

Tipovi paralelizma

- **Paralelizam na nivou bitova** (engl. *bit-level parallelism*)
- **Paralelizam na nivou instrukcija** (engl. *instruction level parallelism* – ILP)

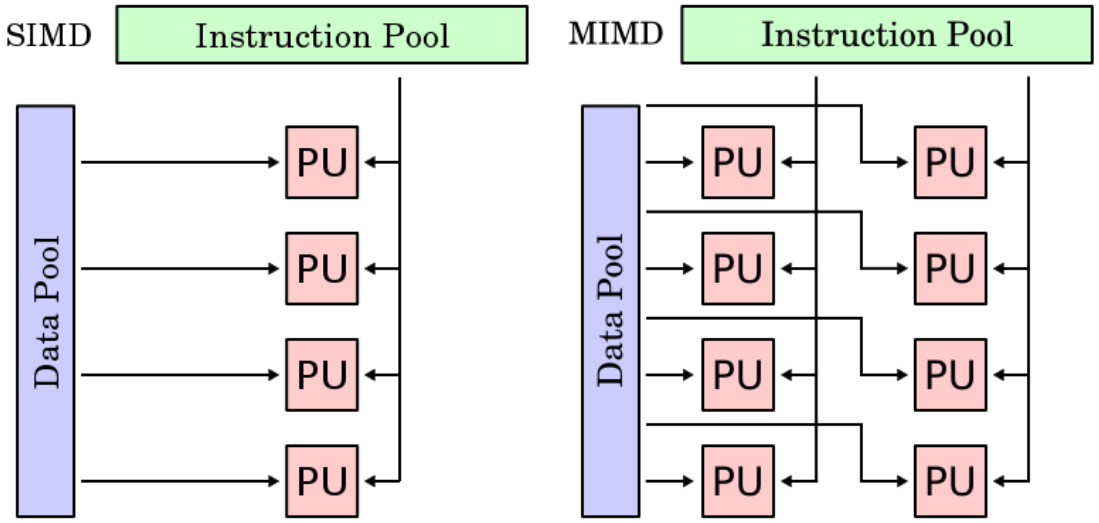


Izvor: https://en.wikipedia.org/wiki/Parallel_computing

- **Paralelizam na nivou zadataka** (engl. *task-level parallelism*)

Paralelizam na nivou podataka i zadataka

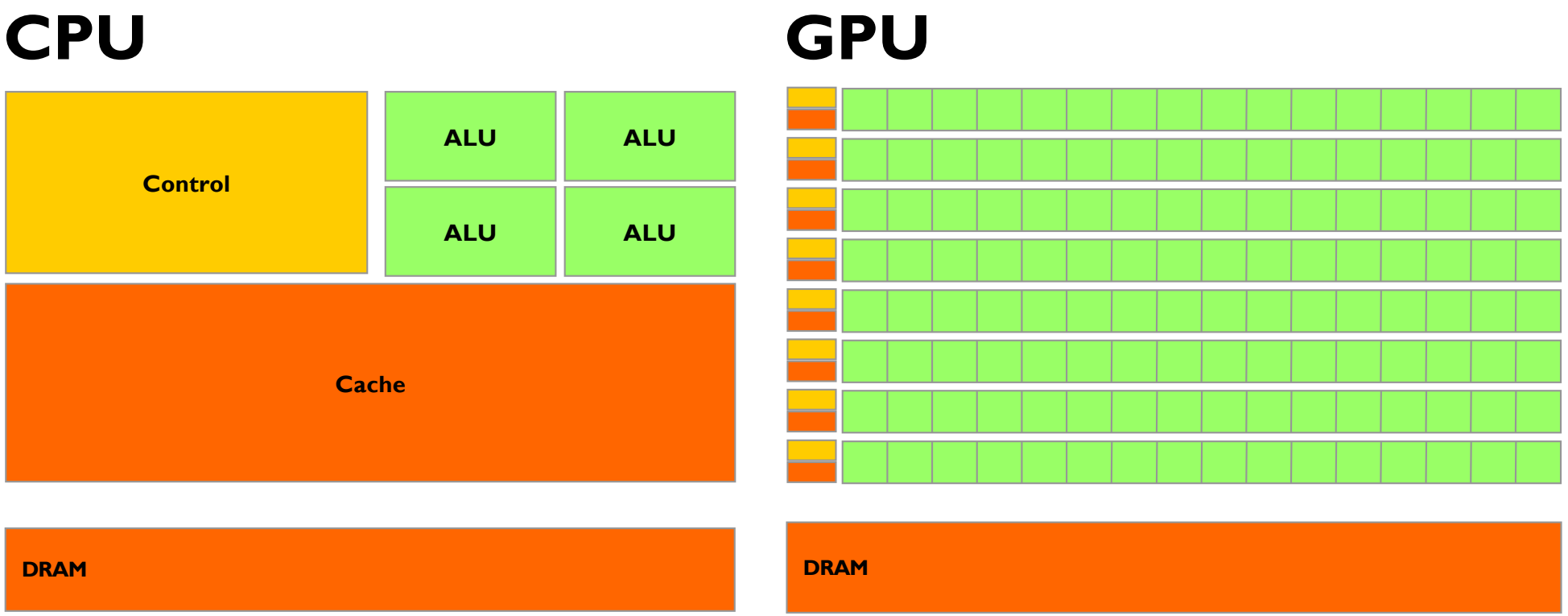
- **Paralelizam na nivou podataka** (engl. *data parallelism*) – tipično za SIMD arhitekture
- **Paralelizam na nivou zadataka** (engl. *task parallelism*)
- **Upleteni paralelizam** (engl. *braided parallelism*) – kombinovani paralelizam na nivoima podataka i zadataka



Heterogeni paralelni računarski sistemi

- Prelazak na **heterogene računarske procesore** je velika promena u **arhitekturama računara** i **računarstvu uopšte**
- **Homogeni računarski sistemi** – jedan ili više procesora iste arhitekture koriste se za izvršavanje programa
- **Heterogeni računarski sistemi** – skup procesora zasnovanih na različitih arhitekturama (CPU, GPU, FPGA, DSP) koristi se za izvršavanje programa
- **Svaki procesor** namenjen je za **različite zadatke** i s toga je njegova arhitektura zasnovana na **različitoj projektnoj filozofiji**
- Izvršavanje zadataka na arhitekturama koje su im najbolje prilagođene vodi do **unapređenih performansi** u smislu vremena i energije**,** ali zahteva **nove tehnike u programiranju** (primer – GPGPU programiranje)

Paralelna obrada na CPU i GPU



von Neumann, višejezgarna

SIMD, mnogojezgarna

Izvor: https://commons.wikimedia.org/wiki/File:Cpu-gpu.svg