Uvodu u informatiku – Hardver savremenih računara

Danijela Simić

hardver

1. oktobar 2025.

Sadržaj i

1. Uvod

2. Procesor (CPU)

3. Memorijska hijerarhija

4. Sistemska magistrala i komunikacija komponenti

5. Ulazno–izlazni podsistem

Uvod

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute
- Cevovod (pipelining) i hazardi

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute
- Cevovod (pipelining) i hazardi
- Paralelizam na nivou instrukcija (ILP) i više jezgara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute
- Cevovod (pipelining) i hazardi
- Paralelizam na nivou instrukcija (ILP) i više jezgara
- GPU i specijalizovani akceleratori (TPU, NPU, DNN)

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute
- Cevovod (pipelining) i hazardi
- Paralelizam na nivou instrukcija (ILP) i više jezgara
- GPU i specijalizovani akceleratori (TPU, NPU, DNN)

Hardver savremenih računara

Šta radimo danas?

- Podsećanje: osnovne komponente savremenog računara
- Procesor (CPU): ALU, kontrolna jedinica, registri
- Ciklus instrukcije: fetch – decode – execute
- Cevovod (pipelining) i hazardi
- Paralelizam na nivou instrukcija (ILP) i više jezgara
- GPU i specijalizovani akceleratori (TPU, NPU, DNN)

Ne pričamo o „delovima računara“ samo spolja (kućište, tastatura), već o unutrašnjoj organizaciji koja određuje performanse savremenih sistema.

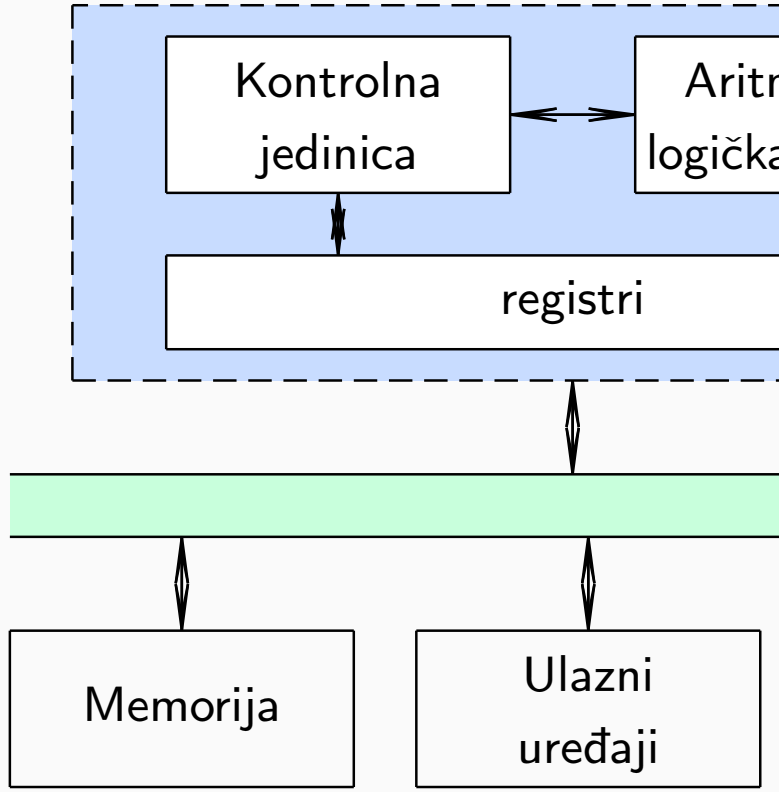
Procesor (CPU)

Procesor (CPU)

Osnovne komponente procesora

Osnovne komponente računara

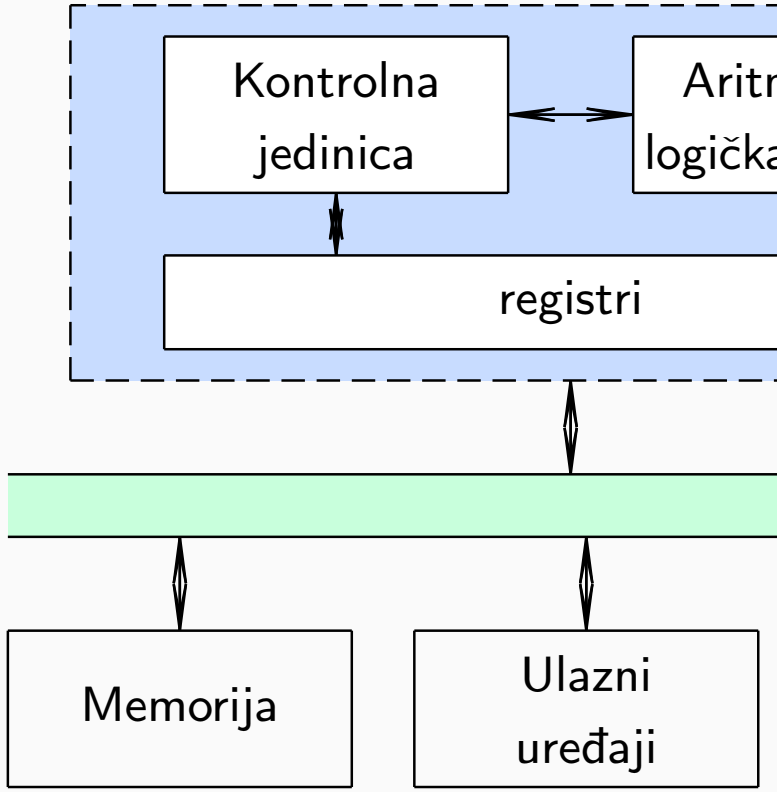
- **Procesor (CPU)** – izvršava instrukcije nad podacima



Sve komponente su povezane i stalno razmenjuju podatke

Osnovne komponente računara

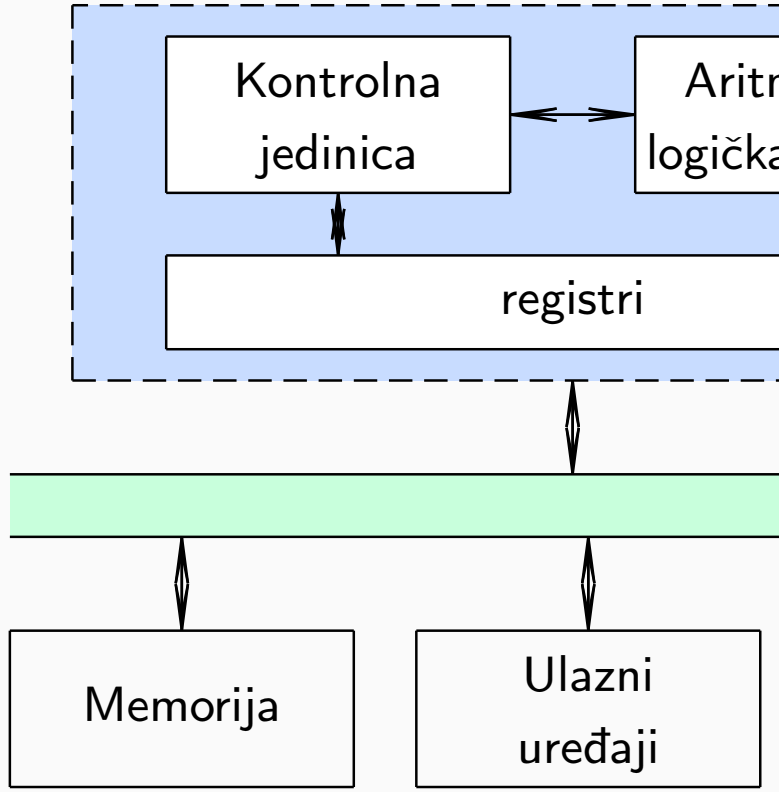
- **Procesor (CPU)** – izvršava instrukcije nad podacima
- **Glavna memorija (RAM)** – čuva programe i podatke



Sve komponente su povezane i stalno razmenjuju podatke

Osnovne komponente računara

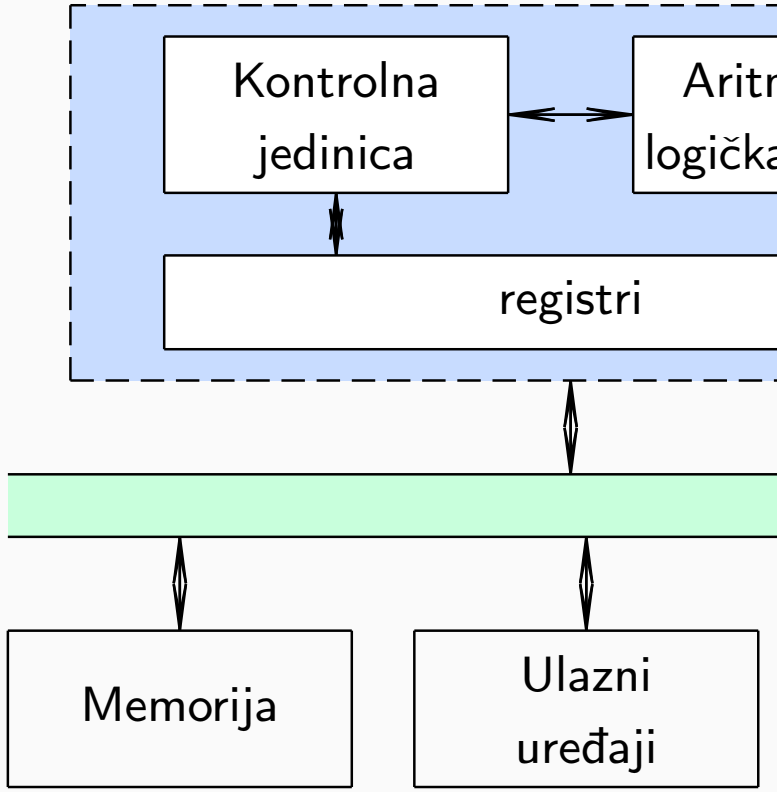
- **Procesor (CPU)** – izvršava instrukcije nad podacima
- **Glavna memorija (RAM)** – čuva programe i podatke
- **Ulazno–izlazni uređaji** – tastatura, ekran, diskovi, mreža



Sve komponente su povezane i stalno razmenjuju podatke

Osnovne komponente računara

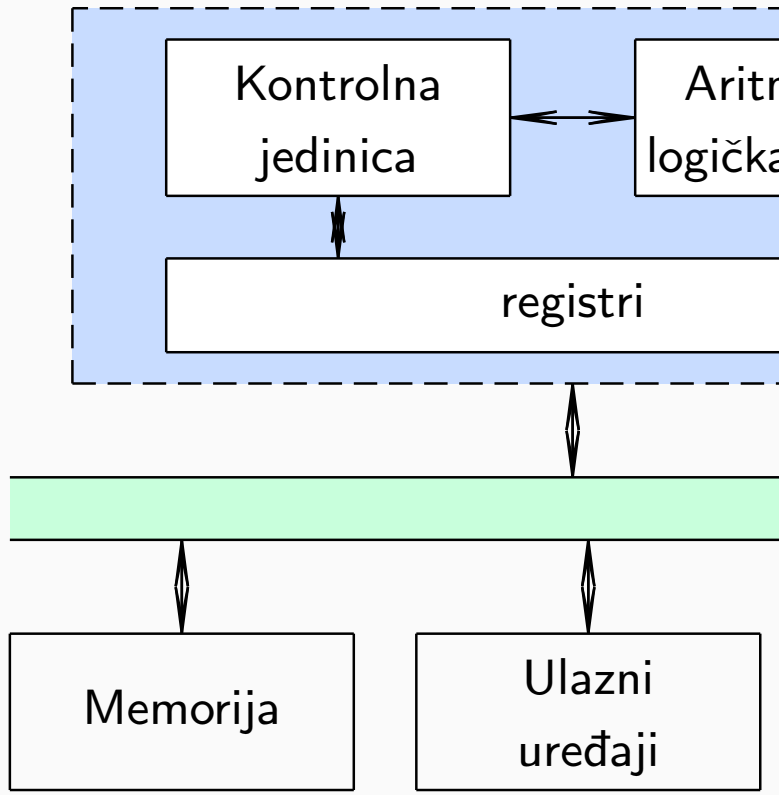
- **Procesor (CPU)** – izvršava instrukcije nad podacima
- **Glavna memorija (RAM)** – čuva programe i podatke
- **Ulazno–izlazni uređaji** – tastatura, ekran, diskovi, mreža
- **Magistrale i kontroleri** – povezuju sve komponente



Sve komponente su povezane i stalno razmenjuju podatke

Osnovne komponente računara

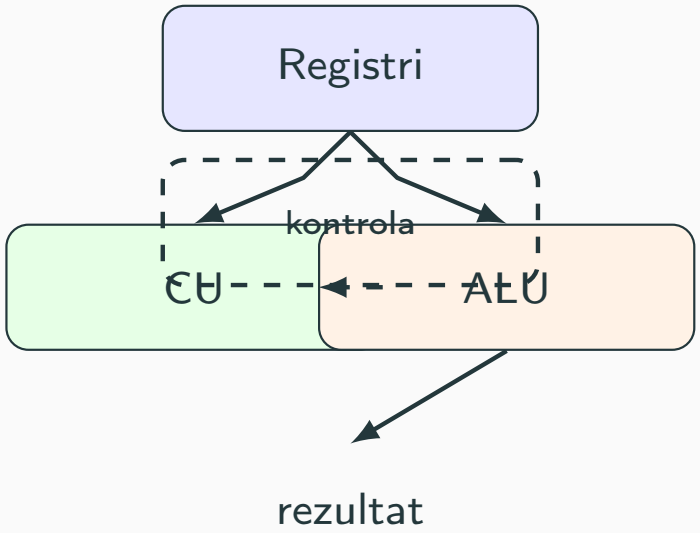
- **Procesor (CPU)** – izvršava instrukcije nad podacima
- **Glavna memorija (RAM)** – čuva programe i podatke
- **Ulazno–izlazni uređaji** – tastatura, ekran, diskovi, mreža
- **Magistrale i kontroleri** – povezuju sve komponente
- **Matična ploča** – fizička integracija sistema



Sve komponente su povezane i stalno razmenjuju podatke

CPU kao centralna procesorska jedinica

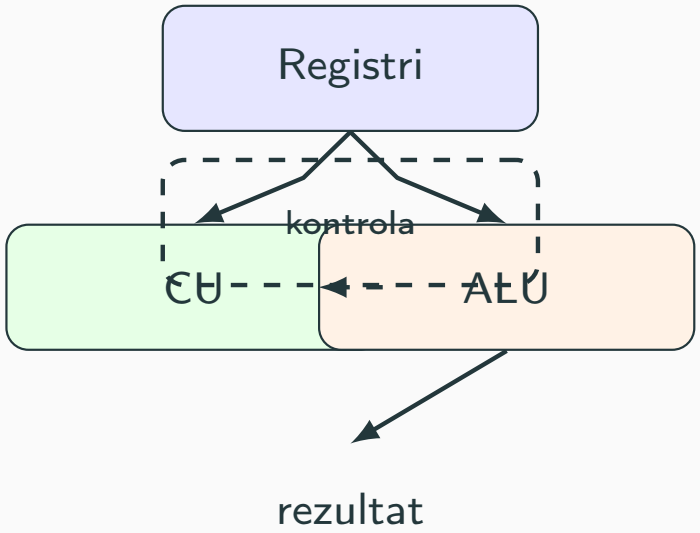
- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

CPU kao centralna procesorska jedinica

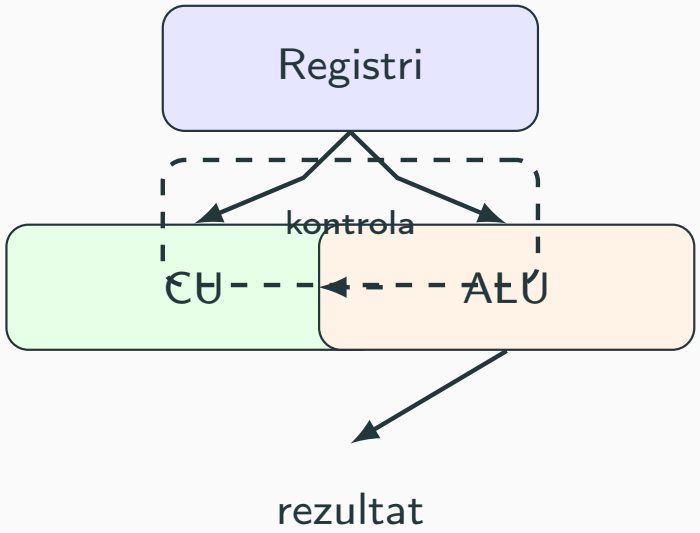
- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu
- Sadrži:



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

CPU kao centralna procesorska jedinica

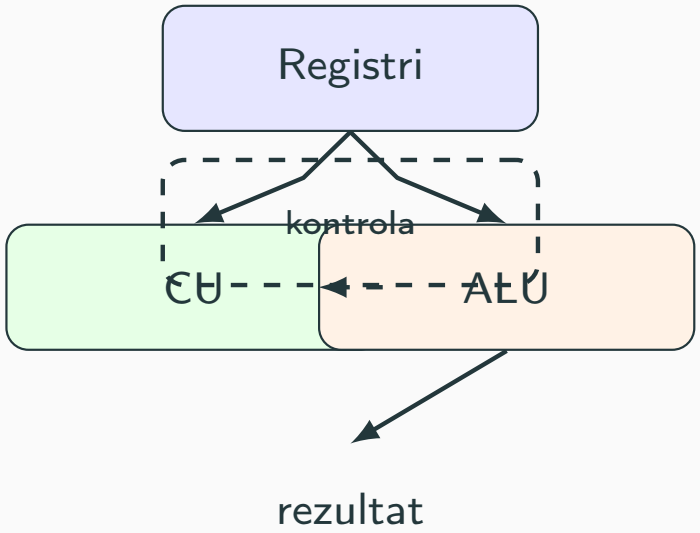
- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu
- Sadrži:
  - **ALU** – aritmetičko-logička jedinica



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

CPU kao centralna procesorska jedinica

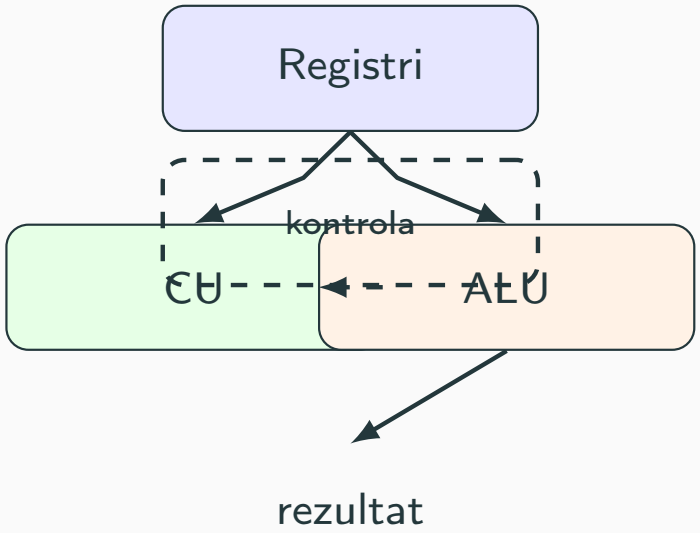
- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu
- Sadrži:
  - **ALU** – aritmetičko-logička jedinica
  - **Kontrolnu jedinicu (CU)**



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

CPU kao centralna procesorska jedinica

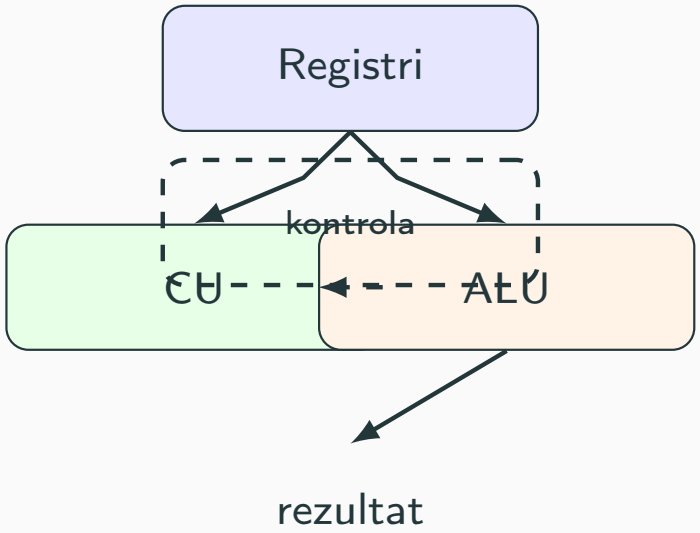
- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu
- Sadrži:
  - **ALU** – aritmetičko-logička jedinica
  - **Kontrolnu jedinicu (CU)**
  - **Registre** – mali, brzi memorijski elementi



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

CPU kao centralna procesorska jedinica

- CPU je implementiran kao mikroprocesor na jednom integrisanom kolu
- Sadrži:
  - **ALU** – aritmetičko-logička jedinica
  - **Kontrolnu jedinicu (CU)**
  - **Registre** – mali, brzi memorijski elementi
- Savremeni procesori imaju više jezgara i složenu internu mikroarhitekturu



CPU = „mašina za instrukcije“ nad registrima i podacima u memoriji.

ALU, kontrolna jedinica i registri

Tri osnovne celine CPU-a

- **ALU (Arithmetic Logic Unit)** sabiranje, oduzimanje, množenje, deljenje, logičke operacije, poređenja

ALU, kontrolna jedinica i registri

Tri osnovne celine CPU-a

- **ALU (Arithmetic Logic Unit)** sabiranje, oduzimanje, množenje, deljenje, logičke operacije, poređenja
- **Kontrolna jedinica (Control Unit)** dekodira instrukcije, generiše upravljačke signale, upravlja tokom podataka

ALU, kontrolna jedinica i registri

Tri osnovne celine CPU-a

- **ALU (Arithmetic Logic Unit)** sabiranje, oduzimanje, množenje, deljenje, logičke operacije, poređenja
- **Kontrolna jedinica (Control Unit)** dekodira instrukcije, generiše upravljačke signale, upravlja tokom podataka
- **Registri** mali broj vrlo brzih memorijskih ćelija fiksne širine

ALU, kontrolna jedinica i registri

Tri osnovne celine CPU-a

- **ALU (Arithmetic Logic Unit)** sabiranje, oduzimanje, množenje, deljenje, logičke operacije, poređenja
- **Kontrolna jedinica (Control Unit)** dekodira instrukcije, generiše upravljačke signale, upravlja tokom podataka
- **Registri** mali broj vrlo brzih memorijskih ćelija fiksne širine

ALU, kontrolna jedinica i registri

Tri osnovne celine CPU-a

- **ALU (Arithmetic Logic Unit)** sabiranje, oduzimanje, množenje, deljenje, logičke operacije, poređenja
- **Kontrolna jedinica (Control Unit)** dekodira instrukcije, generiše upravljačke signale, upravlja tokom podataka
- **Registri** mali broj vrlo brzih memorijskih ćelija fiksne širine

Što je više podataka moguće držati u registrima, to je manje pristupa glavnoj memoriji i sistem je brži.

Procesor (CPU)

Cevovod i tehnike ubrzanja

Ciklus instrukcije

- **Fetch** – dohvatanje instrukcije iz memorije

Ciklus instrukcije

- **Fetch** – dohvatanje instrukcije iz memorije
- **Decode** – dekodiranje i priprema operanada

Ciklus instrukcije

- **Fetch** – dohvatanje instrukcije iz memorije
- **Decode** – dekodiranje i priprema operanada
- **Execute** – izvršavanje operacije u ALU / pristup memoriji

Ciklus instrukcije

- **Fetch** – dohvatanje instrukcije iz memorije
- **Decode** – dekodiranje i priprema operanada
- **Execute** – izvršavanje operacije u ALU / pristup memoriji

Ciklus instrukcije

- **Fetch** – dohvatanje instrukcije iz memorije
- **Decode** – dekodiranje i priprema operanada
- **Execute** – izvršavanje operacije u ALU / pristup memoriji

U svakoj fazi CPU koristi različite delove hardvera (magistralu, registre, ALU...), tako da je prirodno da pokušamo da se faze preklapaju kod više instrukcija (cevovod).

Primer: ciklus instrukcije

Pretpostavimo da su u registrima:

- `R1 = 5`
- `R2 = 3`

i da imamo instrukciju:

```
ADD R1, R2      ; R1 ← R1 + R2
```

- **Fetch**: CPU dohvata instrukciju `ADD R1, R2` iz memorije

Nakon izvršavanja: `R1 = 8`, `R2 = 3`.

Primer: ciklus instrukcije

Pretpostavimo da su u registrima:

- `R1 = 5`
- `R2 = 3`

i da imamo instrukciju:

```
ADD R1, R2      ; R1 ← R1 + R2
```

- **Fetch**: CPU dohvata instrukciju `ADD R1, R2` iz memorije
- **Decode**: CU prepoznaje operaciju sabiranja, čita R1 i R2

Nakon izvršavanja: `R1 = 8`, `R2 = 3`.

Primer: ciklus instrukcije

Pretpostavimo da su u registrima:

- `R1 = 5`
- `R2 = 3`

i da imamo instrukciju:

```
ADD R1, R2      ; R1 ← R1 + R2
```

- **Fetch**: CPU dohvata instrukciju `ADD R1, R2` iz memorije
- **Decode**: CU prepoznaje operaciju sabiranja, čita R1 i R2
- **Execute**: ALU računa $5 + 3 = 8$ i rezultat upisuje nazad u R1

Nakon izvršavanja: `R1 = 8`, `R2 = 3`.

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

```
1) LOAD R1, [A]      ; A = 10
2) LOAD R2, [B]      ; B = 20
3) ADD  R3, R1, R2
4) STORE [C], R3
```

- Pretpostavimo da su u memoriji: `A = 10`, `B = 20`.

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

```
1) LOAD R1, [A]      ; A = 10
2) LOAD R2, [B]      ; B = 20
3) ADD  R3, R1, R2
4) STORE [C], R3
```

- Pretpostavimo da su u memoriji: `A = 10`, `B = 20`.
- Pitanje: koje vrednosti će biti u registrima `R1`, `R2`, `R3` i na adresi `C` nakon izvršavanja?

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

```
1) LOAD R1, [A]      ; A = 10
2) LOAD R2, [B]      ; B = 20
3) ADD  R3, R1, R2
4) STORE [C], R3
```

- Pretpostavimo da su u memoriji: `A = 10`, `B = 20`.
- Pitanje: koje vrednosti će biti u registrima `R1`, `R2`, `R3` i na adresi `C` nakon izvršavanja?
- Kako izgleda izvršavanje – nacrtati na tabli i napisati šta se koristi (CPU, ALU ili registri)?

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

```
1) LOAD R1, [A]      ; A = 10
2) LOAD R2, [B]      ; B = 20
3) ADD  R3, R1, R2
4) STORE [C], R3
```

- Pretpostavimo da su u memoriji: `A = 10`, `B = 20`.
- Pitanje: koje vrednosti će biti u registrima `R1`, `R2`, `R3` i na adresi `C` nakon izvršavanja?
- Kako izgleda izvršavanje – nacrtati na tabli i napisati šta se koristi (CPU, ALU ili registri)?

Mini-zadatak: registri i instrukcije

Data je sledeća sekvenca:

```
1) LOAD R1, [A]      ; A = 10
2) LOAD R2, [B]      ; B = 20
3) ADD  R3, R1, R2
4) STORE [C], R3
```

- Pretpostavimo da su u memoriji: `A = 10`, `B = 20`.
- Pitanje: koje vrednosti će biti u registrima `R1`, `R2`, `R3` i na adresi `C` nakon izvršavanja?
- Kako izgleda izvršavanje – nacrtati na tabli i napisati šta se koristi (CPU, ALU ili registri)?

`R1 = 10`, `R2 = 20`, `R3 = 30`, `C = 30`.

Motivacija za cevovod (pipelining)

Problem sekvencijalnog izvršavanja

- Ako instrukcije izvršavamo „jednu po jednu“, svaka mora da prođe sve faze pre nego što sledeća počne.

Motivacija za cevovod (pipelining)

Problem sekvencijalnog izvršavanja

- Ako instrukcije izvršavamo „jednu po jednu“, svaka mora da prođe sve faze pre nego što sledeća počne.
- Za N instrukcija i k faza to znači oko $k \cdot N$ ciklusa.

Motivacija za cevovod (pipelining)

Problem sekvencijalnog izvršavanja

- Ako instrukcije izvršavamo „jednu po jednu“, svaka mora da prođe sve faze pre nego što sledeća počne.
- Za N instrukcija i k faza to znači oko $k \cdot N$ ciklusa.
- Tokom tog vremena, mnogi delovi procesora su neaktivni.

Motivacija za cevovod (pipelining)

Problem sekvencijalnog izvršavanja

- Ako instrukcije izvršavamo „jednu po jednu“, svaka mora da prođe sve faze pre nego što sledeća počne.
- Za N instrukcija i k faza to znači oko $k \cdot N$ ciklusa.
- Tokom tog vremena, mnogi delovi procesora su neaktivni.

Motivacija za cevovod (pipelining)

Problem sekvencijalnog izvršavanja

- Ako instrukcije izvršavamo „jednu po jednu“, svaka mora da prođe sve faze pre nego što sledeća počne.
- Za N instrukcija i k faza to znači oko $k \cdot N$ ciklusa.
- Tokom tog vremena, mnogi delovi procesora su neaktivni.

Preklapanje faza: dok jedna instrukcija dekodira, sledeća se već dohvata, a prethodna se izvršava. Cilj: približiti se jednoj instrukciji po ciklusu.

Jednostavan petostepeni cevovod

- IF – Instruction Fetch (čitanje instrukcije iz memorije)

Jednostavan petostepeni cevovod

- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)

Jednostavan petostepeni cevovod

- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)
- EX – Execute (ALU operacije ili izračunavanje adrese)

Jednostavan petostepeni cevovod

- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)
- EX – Execute (ALU operacije ili izračunavanje adrese)
- MEM – Memory (čitanje/pisanje podataka iz memorije)

Jednostavan petostepeni cevovod

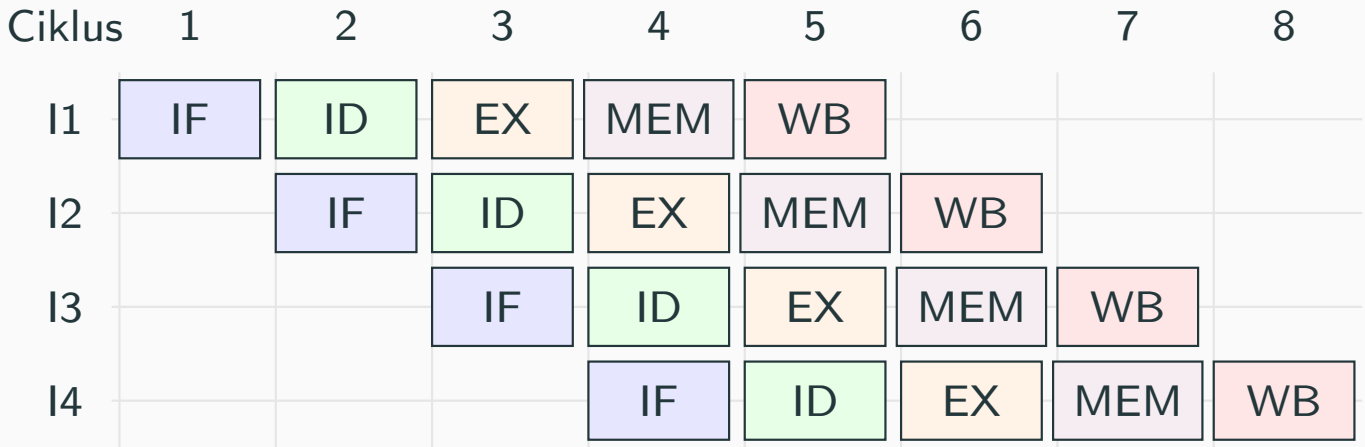
- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)
- EX – Execute (ALU operacije ili izračunavanje adrese)
- MEM – Memory (čitanje/pisanje podataka iz memorije)
- WB – Write Back (upis rezultata u registre)

Jednostavan petostepeni cevovod

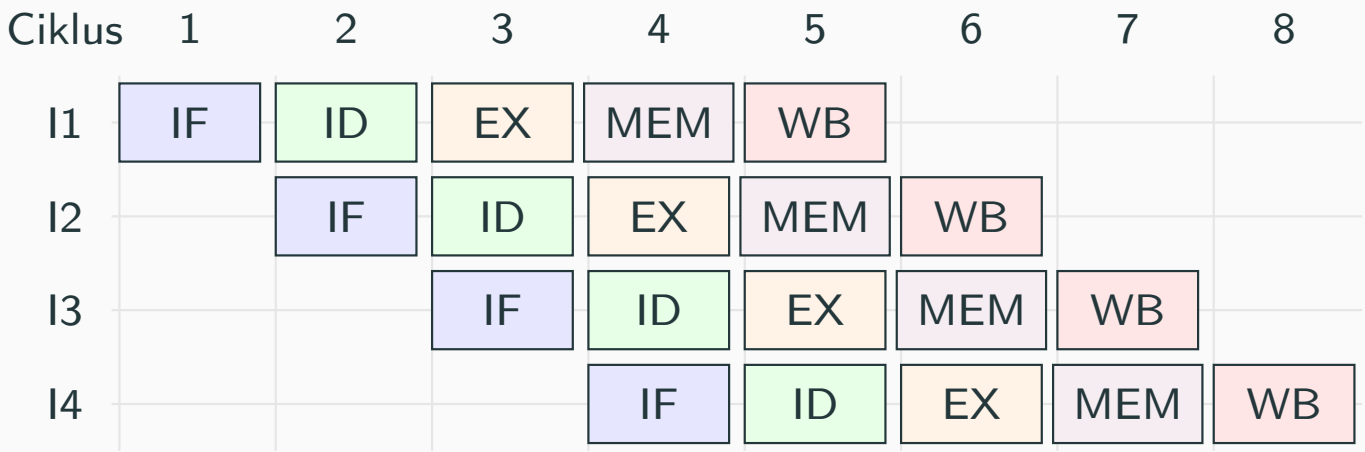
- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)
- EX – Execute (ALU operacije ili izračunavanje adrese)
- MEM – Memory (čitanje/pisanje podataka iz memorije)
- WB – Write Back (upis rezultata u registre)

Jednostavan petostepeni cevovod

- IF – Instruction Fetch (čitanje instrukcije iz memorije)
- ID – Instruction Decode (dekodiranje, čitanje registara)
- EX – Execute (ALU operacije ili izračunavanje adrese)
- MEM – Memory (čitanje/pisanje podataka iz memorije)
- WB – Write Back (upis rezultata u registre)



Mini-zadatak: tabela cevovoda



Poenta je da se u svakom ciklusu izvršava više faza različitih instrukcija istovremeno.

Hazardi u cevovodu

- **Strukturni hazard** – nedovoljno hardverskih resursa

Hazardi u cevovodu

- **Strukturni hazard** – nedovoljno hardverskih resursa
- **Podatkovni hazard** – zavisnosti između instrukcija

Hazardi u cevovodu

- **Strukturni hazard** – nedovoljno hardverskih resursa
- **Podatkovni hazard** – zavisnosti između instrukcija
- **Kontrolni hazard** – grananja i skokovi

Hazardi u cevovodu

- **Strukturni hazard** – nedovoljno hardverskih resursa
- **Podatkovni hazard** – zavisnosti između instrukcija
- **Kontrolni hazard** – grananja i skokovi

Hazardi u cevovodu

- **Strukturni hazard** – nedovoljno hardverskih resursa
- **Podatkovni hazard** – zavisnosti između instrukcija
- **Kontrolni hazard** – grananja i skokovi

Hazardi sprečavaju da sledeća instrukcija uđe u predviđenu fazu u narednom ciklusu $\rightarrow$ javljaju se zadrške (stall) i smanjuje se ubrzanje.

Hazard 1: podatkovni (RAW)

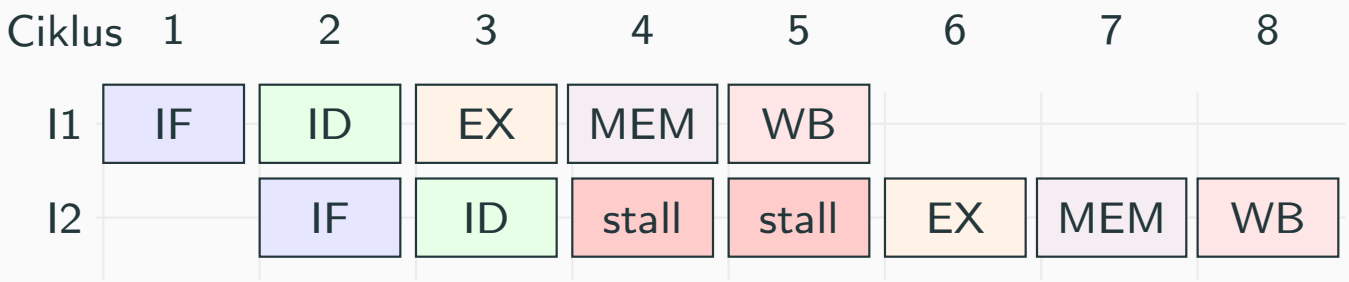
Klasifikujte hazard

```
I1:  a = b + c;
I2:  d = a + 1;
```

Hazard 1: podatkovni (RAW)

Klasifikujte hazard

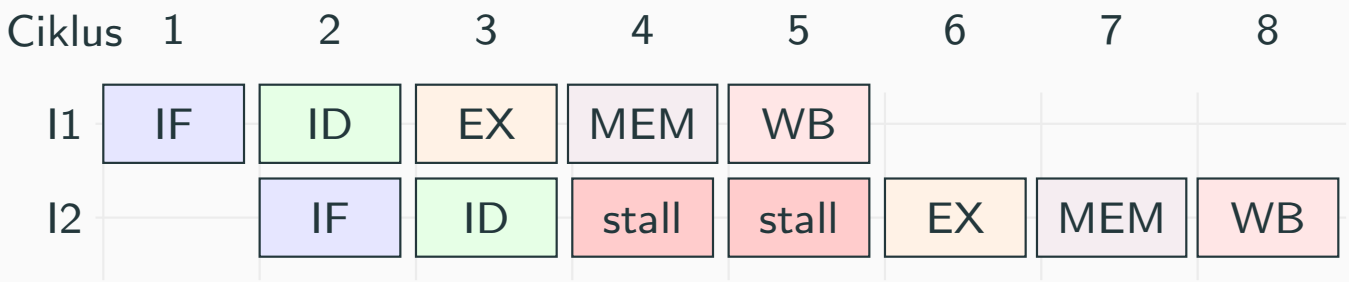
```
I1:  a = b + c;
I2:  d = a + 1;
```



Hazard 1: podatkovni (RAW)

Klasifikujte hazard

```
I1:  a = b + c;
I2:  d = a + 1;
```

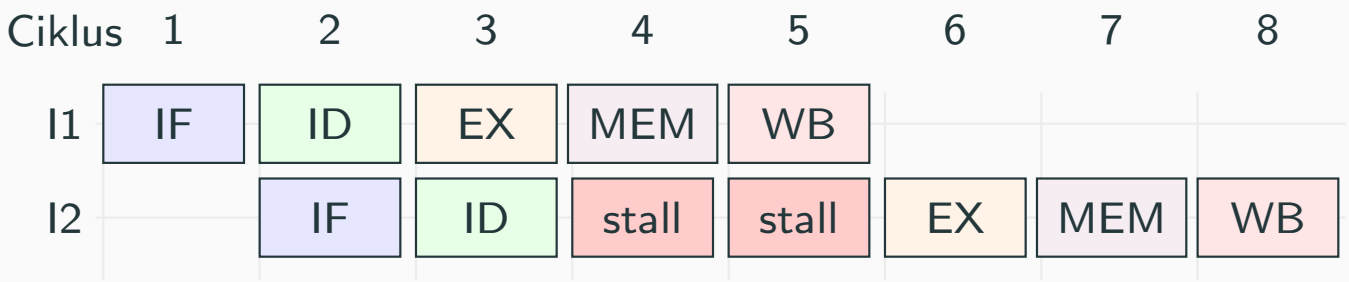


- I2 koristi rezultat instrukcije I1 → podatkovni (RAW) hazard.

Hazard 1: podatkovni (RAW)

Klasifikujte hazard

```
I1:  a = b + c;
I2:  d = a + 1;
```

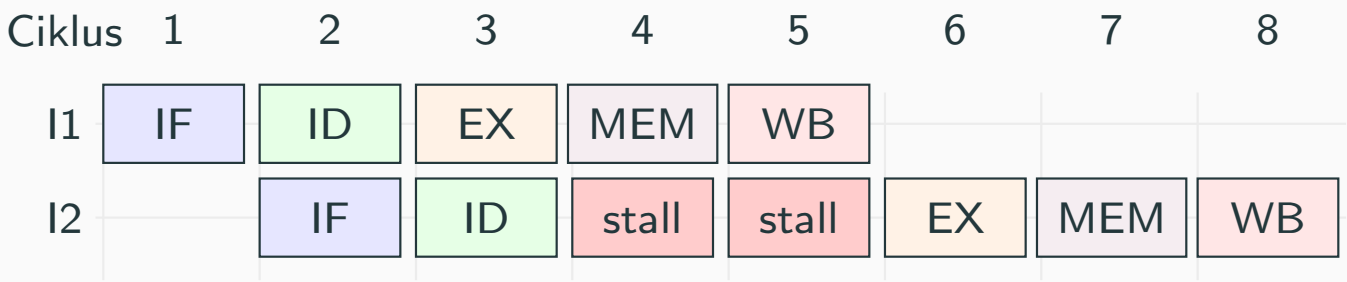


- I2 koristi rezultat instrukcije I1 $\rightarrow$ podatkovni (RAW) hazard.
- Bez prosleđivanja (forwarding), I2 mora da čeka da I1 završi WB fazu.

Hazard 1: podatkovni (RAW)

Klasifikujte hazard

```
I1:  a = b + c;
I2:  d = a + 1;
```



- I2 koristi rezultat instrukcije I1 → podatkovni (RAW) hazard.
- Bez prosleđivanja (forwarding), I2 mora da čeka da I1 završi WB fazu.
- Crvene ćelije označavaju cikluse zastoja (*stall*) u ID fazi.

Hazard 2: kontrolni (grananje)

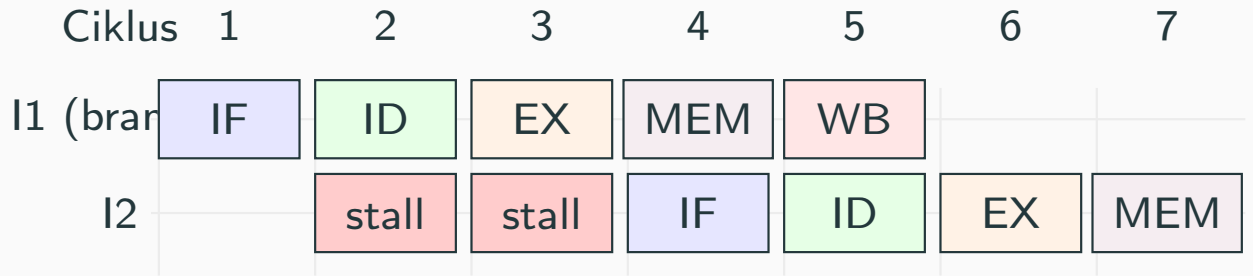
Klasifikujte hazard:

```
I1: if (x > 0)
        y = f(x);
    else
        y = g(x);
I2: ... // sledeća instrukcija, zavisi od grananja
```

Hazard 2: kontrolni (grananje)

Klasifikujte hazard:

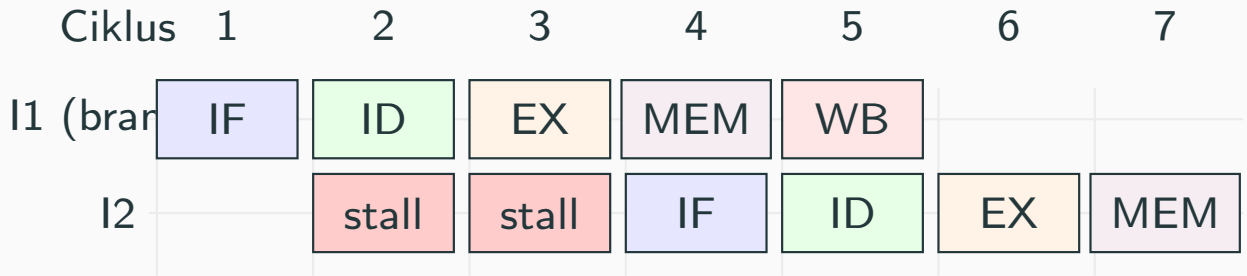
```
I1: if (x > 0)
        y = f(x);
    else
        y = g(x);
I2: ... // sledeća instrukcija, zavisi od grananja
```



Hazard 2: kontrolni (grananje)

Klasifikujte hazard:

```
I1: if (x > 0)
        y = f(x);
    else
        y = g(x);
I2: ... // sledeća instrukcija, zavisi od grananja
```

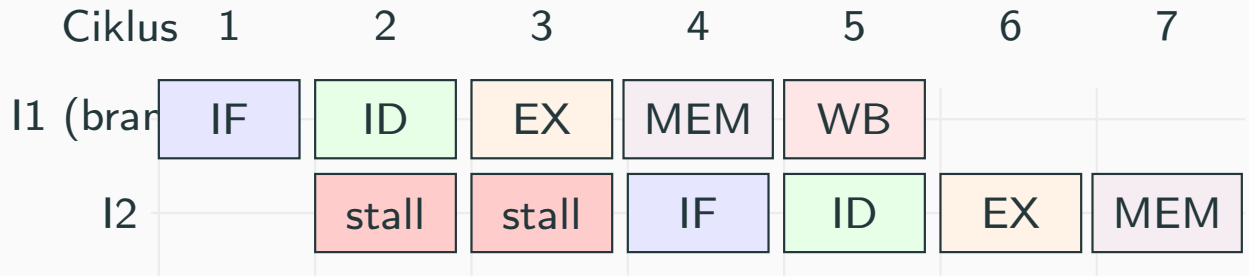


- Ishod grananja (I1) saznaje se tek u EX fazi.

Hazard 2: kontrolni (grananje)

Klasifikujte hazard:

```
I1: if (x > 0)
        y = f(x);
    else
        y = g(x);
I2: ... // sledeća instrukcija, zavisi od grananja
```



- Ishod grananja (I1) saznaje se tek u EX fazi.
- Dok se grananje ne razreši, cevovod ne zna koju instrukciju dalje da učita.

Hazard 3: strukturni (resursi)

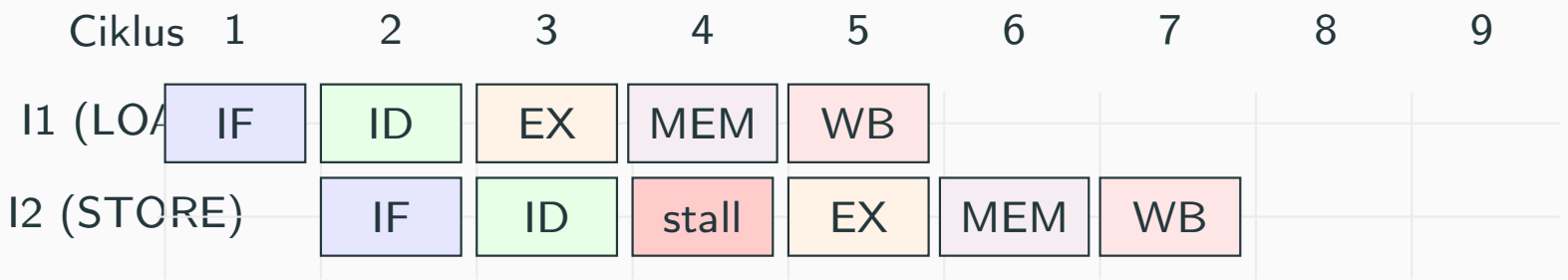
Klasifikujte hazard

```
I1: LOAD  R1, [A]    // instrukcije i podaci u istoj
                     // memoriji
I2: STORE [B], R2
```

Hazard 3: strukturni (resursi)

Klasifikujte hazard

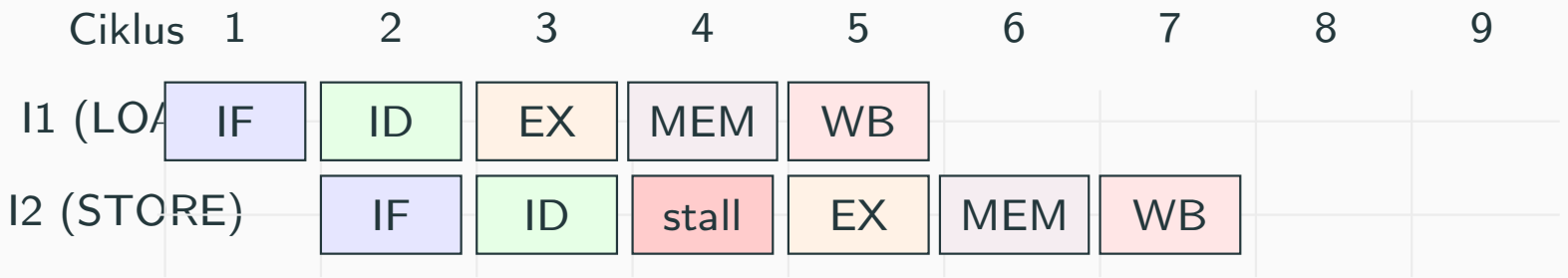
```
I1: LOAD  R1, [A]    // instrukcije i podaci u istoj
                     // memoriji
I2: STORE [B], R2
```



Hazard 3: strukturni (resursi)

Klasifikujte hazard

```
I1: LOAD  R1, [A]    // instrukcije i podaci u istoj
                     // memoriji
I2: STORE [B], R2
```

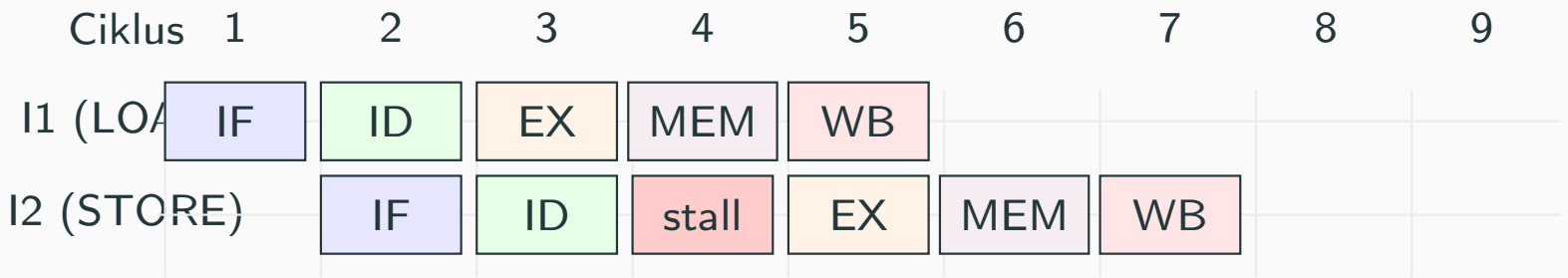


- Jedna zajednička memorija služi i za instrukcije i za podatke.

Hazard 3: strukturni (resursi)

Klasifikujte hazard

```
I1: LOAD  R1, [A]    // instrukcije i podaci u istoj
                     // memoriji
I2: STORE [B], R2
```

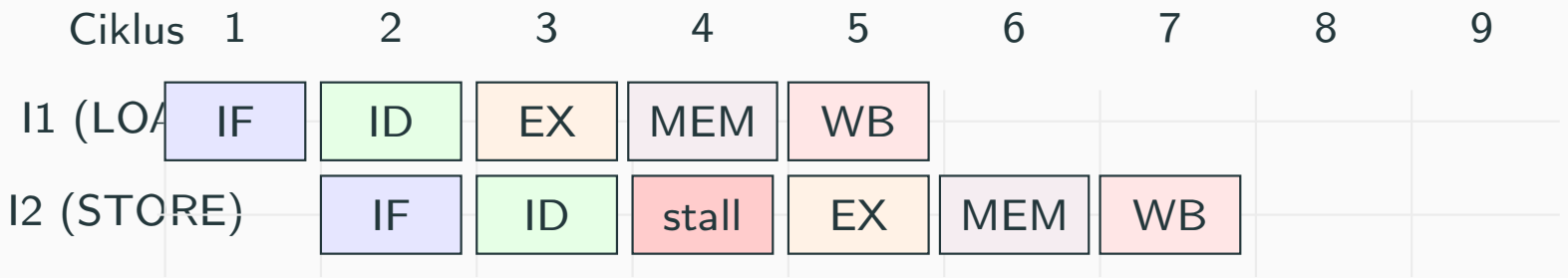


- Jedna zajednička memorija služi i za instrukcije i za podatke.
- Kada I1 koristi MEM fazu (čitajući podatke), I2 ne može istovremeno da radi STORE.

Hazard 3: strukturni (resursi)

Klasifikujte hazard

```
I1: LOAD  R1, [A]    // instrukcije i podaci u istoj
                     // memoriji
I2: STORE [B], R2
```



- Jedna zajednička memorija služi i za instrukcije i za podatke.
- Kada I1 koristi MEM fazu (čitajući podatke), I2 ne može istovremeno da radi STORE.
- Crveni „stall“ u IF I2 $\rightarrow$ strukturni hazard (nedovoljno hardverskih resursa).

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:
  - menjanje redosleda nezavisnih instrukcija

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:
  - menjanje redosleda nezavisnih instrukcija
  - „odmotavanje“ petlji (loop unrolling)

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:
  - menjanje redosleda nezavisnih instrukcija
  - „odmotavanje“ petlji (loop unrolling)
  - dinamičko raspoređivanje i spekulativno izvršavanje

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:
  - menjanje redosleda nezavisnih instrukcija
  - „odmotavanje“ petlji (loop unrolling)
  - dinamičko raspoređivanje i spekulativno izvršavanje

Paralelizam na nivou instrukcija (ILP)

ILP (Instruction-Level Parallelism) označava mogućnost da se više nezavisnih instrukcija izvršava preklopljeno ili istovremeno, bez narušavanja ispravnosti programa.

- Zavisnosti (RAW, WAR, WAW) i kontrolne zavisnosti ograničavaju ILP
- Kompajler i hardver pokušavaju da povećaju ILP:
  - menjanje redosleda nezavisnih instrukcija
  - „odmotavanje“ petlji (loop unrolling)
  - dinamičko raspoređivanje i spekulativno izvršavanje

Što je više nezavisnih instrukcija u osnovnom bloku, to je veći ILP koji arhitektura može da iskoristi.

Predikcija skokova

- Grananja uvode kontrolne hazarde: ne znamo unapred koja instrukcija je sledeća.

Predikcija skokova

- Grananja uvode kontrolne hazarde: ne znamo unapred koja instrukcija je sledeća.
- Predikcija skokova pokušava da „pogodi“ ishod skoka *pre* nego što se on zaista izračuna.

Predikcija skokova

- Grananja uvode kontrolne hazarde: ne znamo unapred koja instrukcija je sledeća.
- Predikcija skokova pokušava da „pogodi“ ishod skoka *pre* nego što se on zaista izračuna.
- Ako je pogodak – cevovod ostaje pun; ako je promašaj – cevovod se prazni i gubimo cikluse.

Predikcija skokova

- Grananja uvode kontrolne hazarde: ne znamo unapred koja instrukcija je sledeća.
- Predikcija skokova pokušava da „pogodi“ ishod skoka *pre* nego što se on zaista izračuna.
- Ako je pogodak – cevovod ostaje pun; ako je promašaj – cevovod se prazni i gubimo cikluse.

Predikcija skokova

- Grananja uvode kontrolne hazarde: ne znamo unapred koja instrukcija je sledeća.
- Predikcija skokova pokušava da „pogodi“ ishod skoka *pre* nego što se on zaista izračuna.
- Ako je pogodak – cevovod ostaje pun; ako je promašaj – cevovod se prazni i gubimo cikluse.

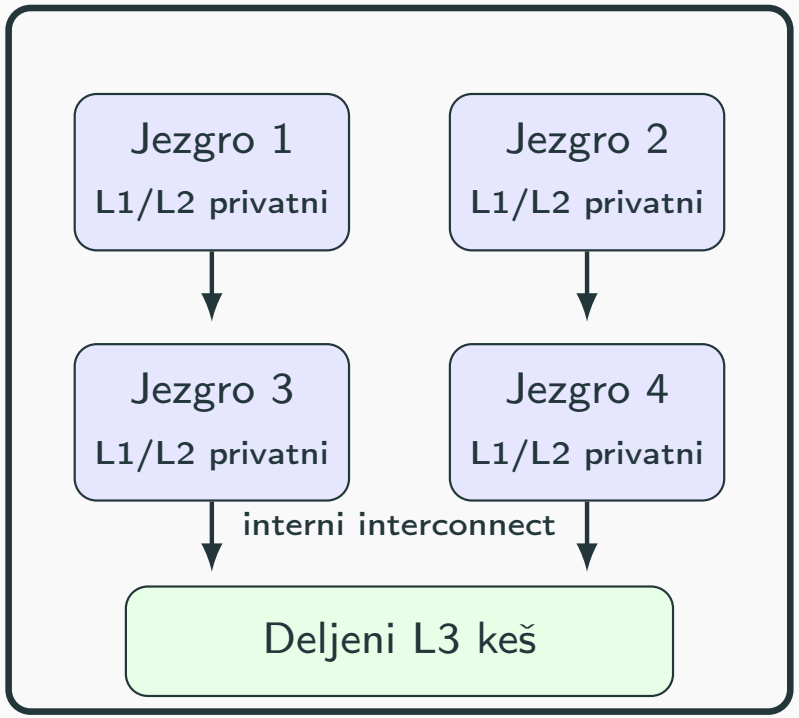
Najjednostavniji model koristi 2-bitni zasićujući brojač po grani; savremeni procesori koriste hibridne i TAGE prediktore sa globalnom istorijom i tagovima za visoku tačnost.

Procesor (CPU)

Procesori sa više jezgara i paralelizam

Više jezgara i TLP

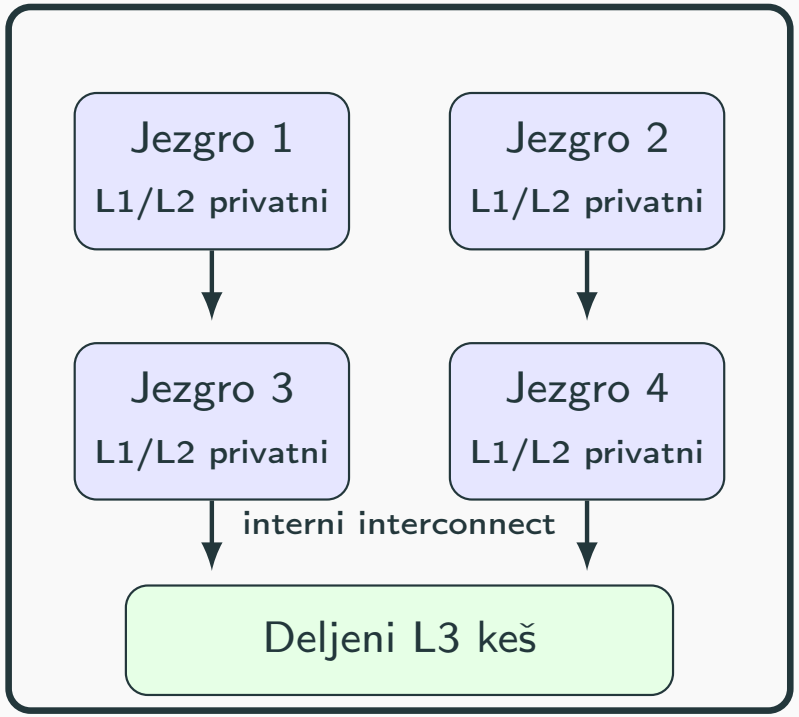
- ILP: paralelizam unutar jedne niti



Savremeni CPU kombinuje ILP (cevovod, višestruko izdavanje) i TLP (više jezgara i niti).

Više jezgara i TLP

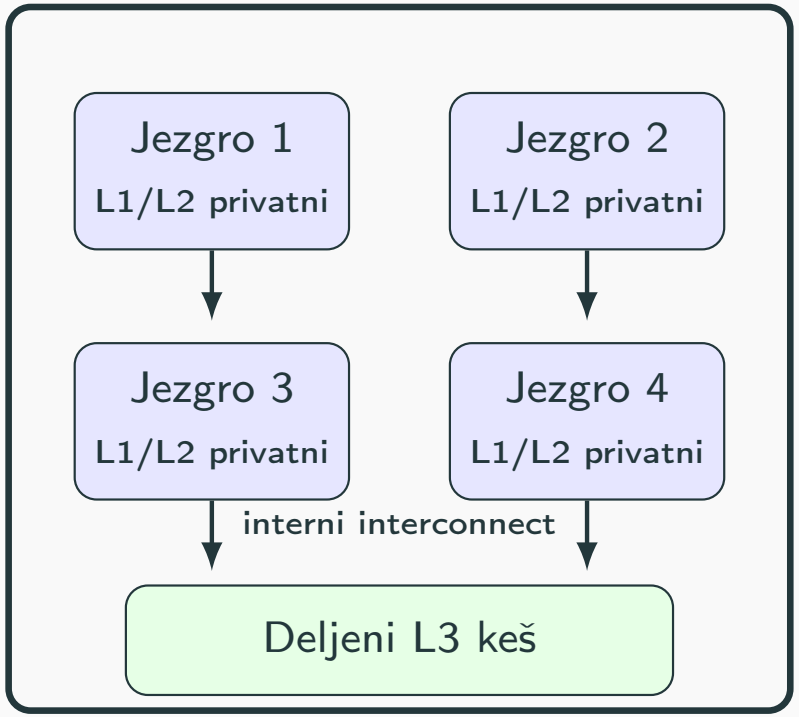
- ILP: paralelizam unutar jedne niti
- TLP (Thread-Level Parallelism): više niti i procesa



Savremeni CPU kombinuje ILP (cevovod, višestruko izdavanje) i TLP (više jezgara i niti).

Više jezgara i TLP

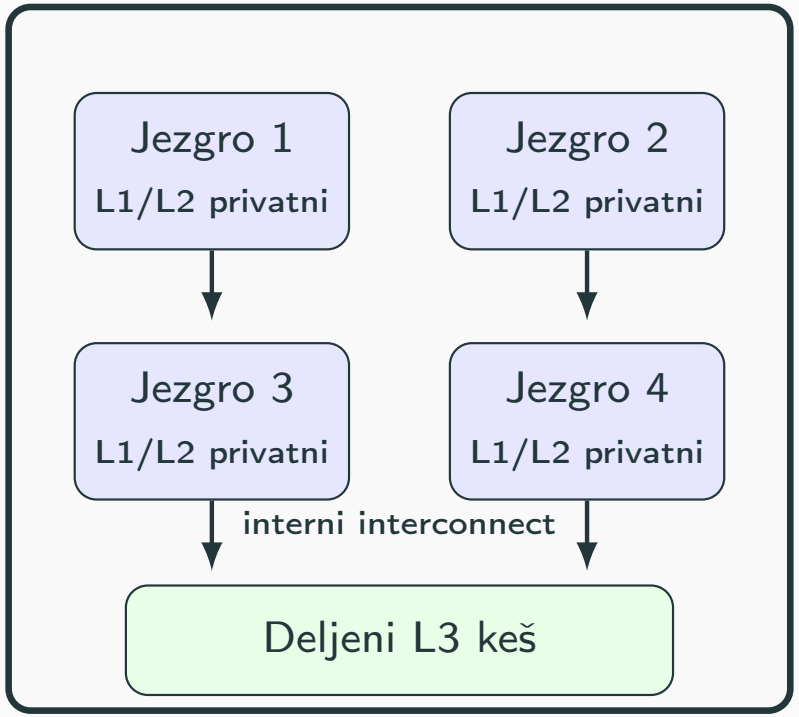
- ILP: paralelizam unutar jedne niti
- TLP (Thread-Level Parallelism): više niti i procesa
- **Multicore** procesori: više jezgara na istom čipu



Savremeni CPU kombinuje ILP (cevovod, višestruko izdavanje) i TLP (više jezgara i niti).

Više jezgara i TLP

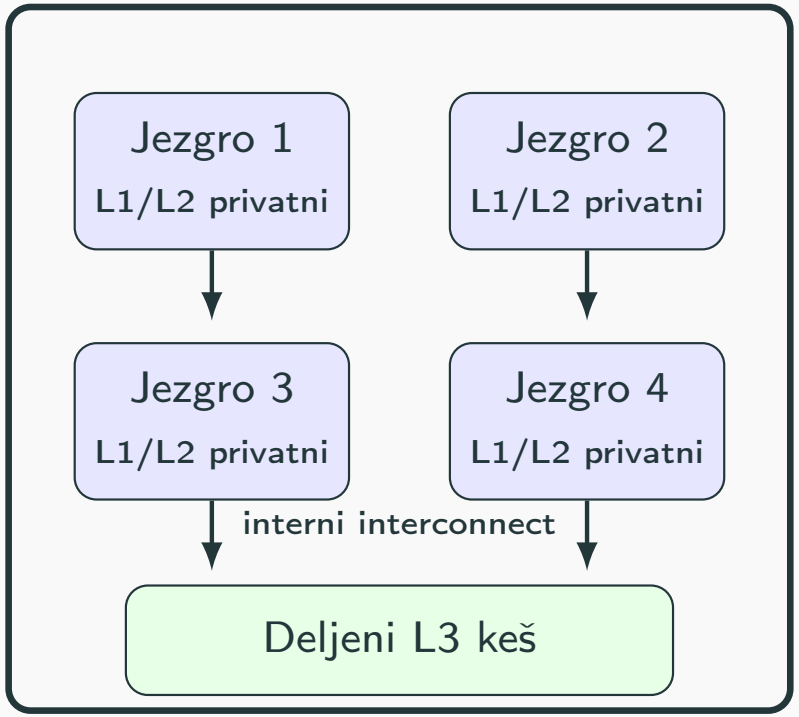
- ILP: paralelizam unutar jedne niti
- TLP (Thread-Level Parallelism): više niti i procesa
- **Multicore** procesori: više jezgara na istom čipu
- Jezgra dele memorijsku hijerarhiju i magistrale



Savremeni CPU kombinuje ILP (cevovod, višestruko izdavanje) i TLP (više jezgara i niti).

Više jezgara i TLP

- ILP: paralelizam unutar jedne niti
- TLP (Thread-Level Parallelism): više niti i procesa
- **Multicore** procesori: više jezgara na istom čipu
- Jezgra dele memorijsku hijerarhiju i magistrale
- Amdalov zakon ograničava ukupno ubrzanje



Savremeni CPU kombinuje ILP (cevovod, višestruko izdavanje) i TLP (više jezgara i niti).

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

- **DNN akceleratori** (TPU, NPU, Tensor Cores)

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

- **DNN akceleratori** (TPU, NPU, Tensor Cores)
- Optimisani za matrične operacije i konvolucije

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

- **DNN akceleratori** (TPU, NPU, Tensor Cores)
- Optimisani za matrične operacije i konvolucije
- Domain-specific architecture: visoke performanse po vatu

GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

- **DNN akceleratori** (TPU, NPU, Tensor Cores)
- Optimisani za matrične operacije i konvolucije
- Domain-specific architecture: visoke performanse po vatu

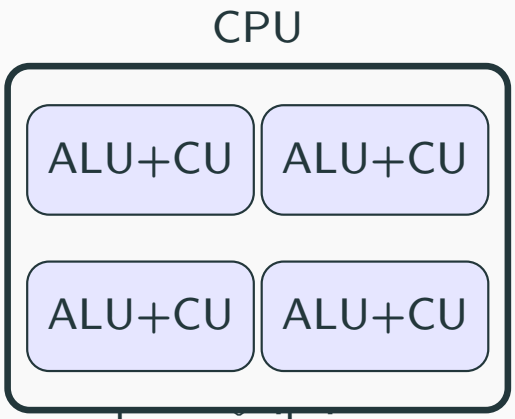
GPU i specijalizovani akceleratori

- **GPU** – stotine ili hiljade jednostavnih jezgara
- Model SIMD/SIMT: ista instrukcija nad velikim brojem podataka
- Idealno za grafiku, naučne računare, mašinsko učenje
- Programski modeli: CUDA, OpenCL

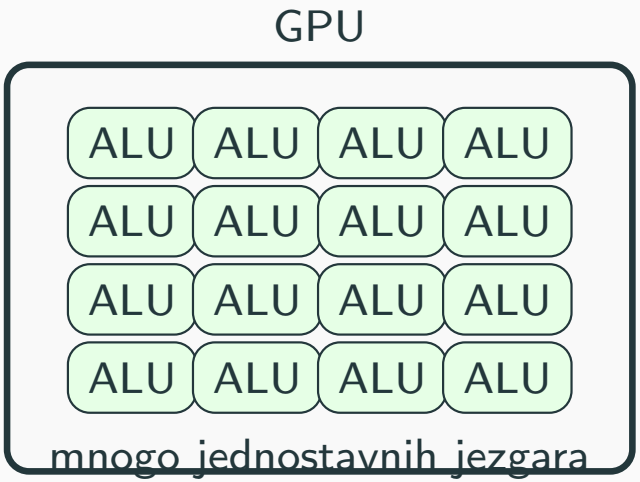
- **DNN akceleratori** (TPU, NPU, Tensor Cores)
- Optimisani za matrične operacije i konvolucije
- Domain-specific architecture: visoke performanse po vatu

Heterogeni sistemi kombinuju CPU + GPU + specijalizovane akceleratore kako bi postigli bolje performanse i energetsku efikasnost.

GPU i specijalizovani akceleratori

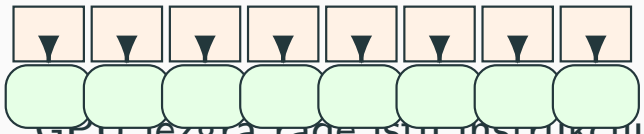


malo snažnih jezgara
(kompleksna jezgra)



mnogo jednostavnih jezgara
(SIMD/SIMT, masivni DLP)

pikseli slike (ulaz)



GPU jezgra rade istu instrukciju
nad različitim pikselima

svako jezgro izvršava istu instrukciju, npr.

```
brightness = brightness + 20;
```

Memorijska hijerarhija

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)
- Glavna memorija (DRAM) i „zid memorije”

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)
- Glavna memorija (DRAM) i „zid memorije”
- SSD i nevolatilne memorije (NVM)

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)
- Glavna memorija (DRAM) i „zid memorije”
- SSD i nevolatilne memorije (NVM)
- Virtuelna memorija – iluzija velikog, kontinualnog prostora

Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)
- Glavna memorija (DRAM) i „zid memorije”
- SSD i nevolatilne memorije (NVM)
- Virtuelna memorija – iluzija velikog, kontinualnog prostora

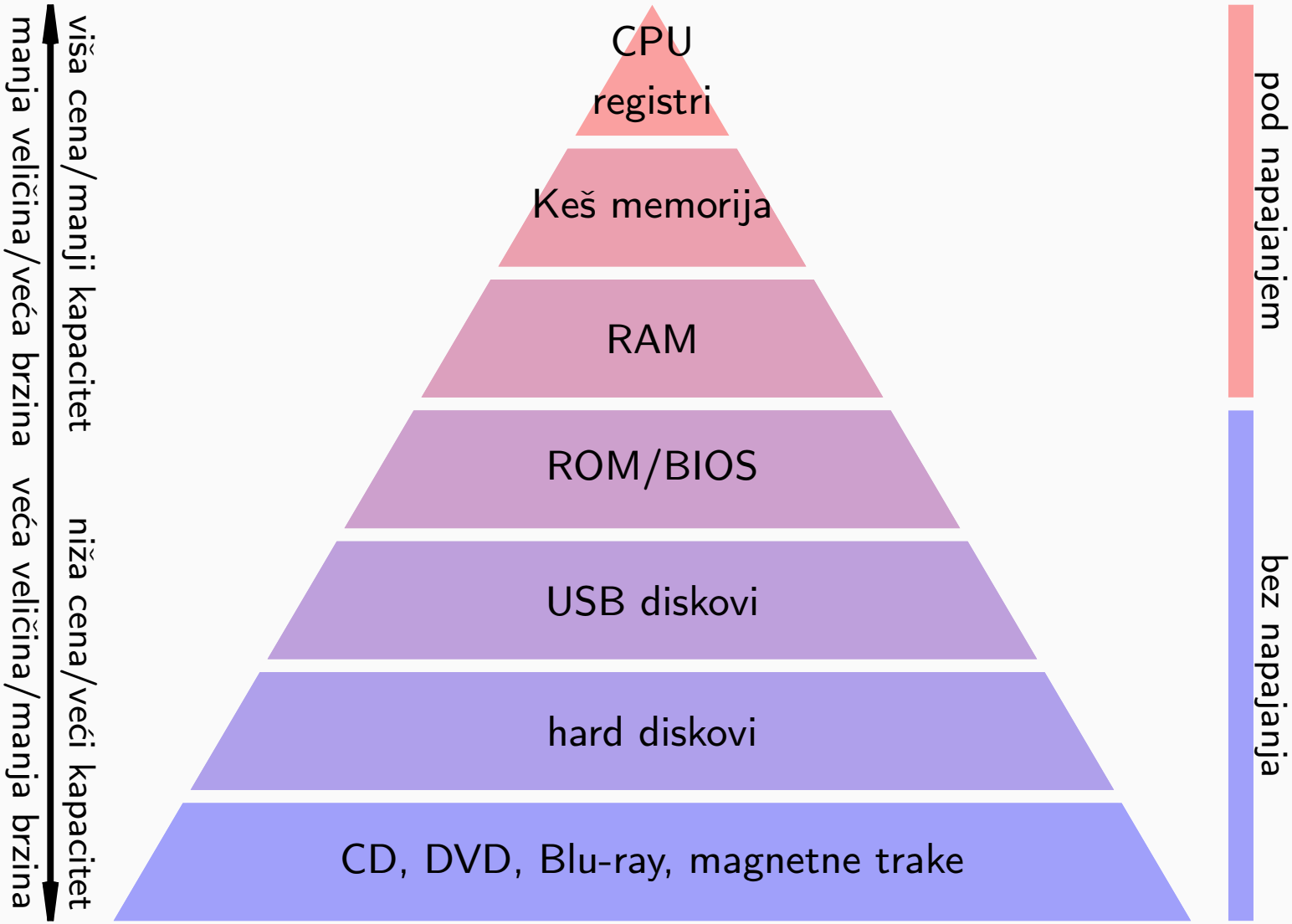
Cilj časa

Šta radimo danas?

- Memorijska hijerarhija: od registara do SSD-a
- Registri, keš memorije (`L1`–`L3`)
- Glavna memorija (DRAM) i „zid memorije”
- SSD i nevolatilne memorije (NVM)
- Virtuelna memorija – iluzija velikog, kontinualnog prostora

Razumeti zašto postoji hijerarhija memorije i kako različiti nivoi balansiraju brzinu, kapacitet, cenu i energiju.

Memorijska hijerarhija



Memorijska hijerarhija – šta mislite?

Popunite tabelu: kako se menjaju **brzina**, **kapacitet** i **cena po bitu** kada se spuštamo niz hijerarhiju memorije?

Nivo	**Brzina**	**Kapacitet**	**Cene (2025.)**
Registri	?	?	?
L1	?	?	?
L2	?	?	?
RAM	?	?	?
SSD	?	?	?

Ideja: *gore* u hijerarhiji je brzo i malo, *dole* je sporo i ogromno. Gde su najskuplji bitovi?

Memorijska hijerarhija – konkretne vrednosti (2025)

Nivo	Brzina	Tipičan kapacitet	Cena (2025)
Registri	0.2–0.5 ns	1–10 KB	$\sim$ \$500–\$1000 po MB (S
L1 keš	1–4 ns	32–64 KB	$\sim$ \$150–\$300 po MB (S
L2 keš	4–12 ns	256 KB–2 MB	$\sim$ \$20–\$50 po MB (SR
RAM (DDR4/5)	60–100 ns	8–64 GB	\$3–\$5 po GB
SSD (NVMe)	50–150 μs	0.5–4 TB	\$0.05–\$0.10 po GB

Latencija raste za **1000×** od RAM-a ka SSD-u, a cena po bitu opada za preko **10000×** od registara ka SSD-u.

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.
- Ovo ograničenje se naziva „zid memorije“ (memory wall).

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.
- Ovo ograničenje se naziva „zid memorije“ (memory wall).

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.
- Ovo ograničenje se naziva „zid memorije“ (memory wall).

- Uvesti više nivoa bržih, manjih memorija: registre, keševe.

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.
- Ovo ograničenje se naziva „zid memorije“ (memory wall).

- Uvesti više nivoa bržih, manjih memorija: registre, keševe.
- Iskoristiti lokalnost pristupa (vremensku i prostornu).

„Zid memorije“

- Brzina procesora je rasla mnogo brže od brzine glavne memorije.
- CPU bi većinu vremena proveo čekajući podatke iz DRAM-a.
- Ovo ograničenje se naziva „zid memorije“ (memory wall).

- Uvesti više nivoa bržih, manjih memorija: registre, keševe.
- Iskoristiti lokalnost pristupa (vremensku i prostornu).
- U proseku sistem se ponaša skoro kao najbrža memorija u lancu.

Memorijska hijerarhija

Registri i keš memorije (L1–L3)

Registri i latencija

Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a

Registri i latencija

Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a
- Fiksna širina (npr. 32 bita, 64 bita)

Registri i latencija

Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a
- Fiksna širina (npr. 32 bita, 64 bita)
- Latencija: **1 takt**

Registri i latencija

Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a
- Fiksna širina (npr. 32 bita, 64 bita)
- Latencija: **1 takt**
- Implementirani u SRAM tehnologiji (šest tranzistora, bez kondenzatora, nema potrebe za osvežavanjem)

Registri i latencija

Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a
- Fiksna širina (npr. 32 bita, 64 bita)
- Latencija: **1 takt**
- Implementirani u SRAM tehnologiji (šest tranzistora, bez kondenzatora, nema potrebe za osvežavanjem)

Registri i latencija

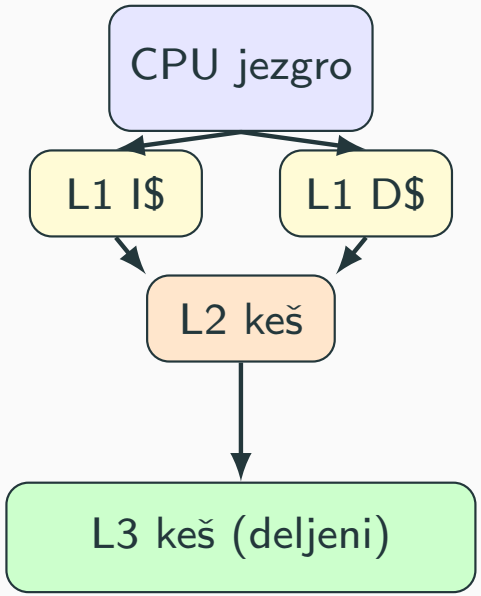
Registri procesora

- Mala, vrlo brza memorija unutar jezgra CPU-a
- Fiksna širina (npr. 32 bita, 64 bita)
- Latencija: **1 takt**
- Implementirani u SRAM tehnologiji (šest tranzistora, bez kondenzatora, nema potrebe za osvežavanjem)

Sve aritmetičke i logičke operacije odvijaju se nad podacima u registrima $\rightarrow$ zato je važno što više relevantnih podataka zadržati na ovom nivou.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a

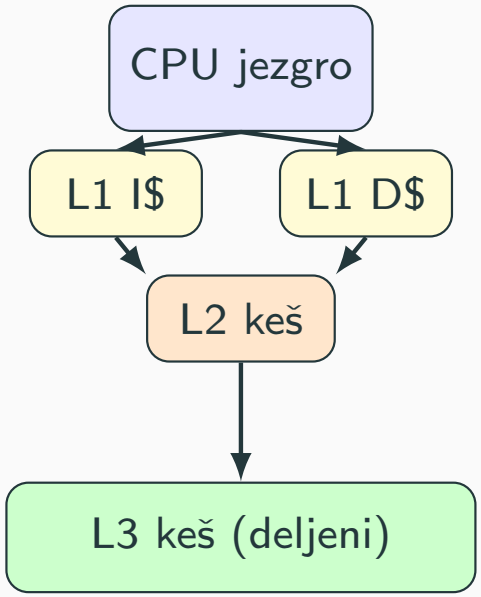


→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: `L1` (najbrži, najmanji), `L2`, `L3`

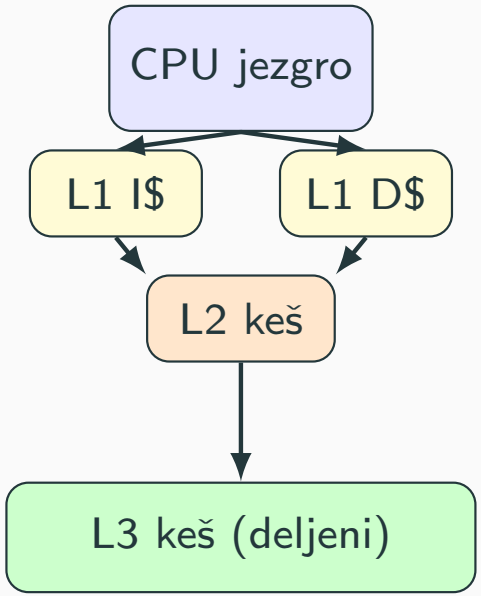


$\rightarrow$ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: `L1` (najbrži, najmanji), `L2`, `L3`
- Čuvaju **kopije** često korišćenih podataka i instrukcija

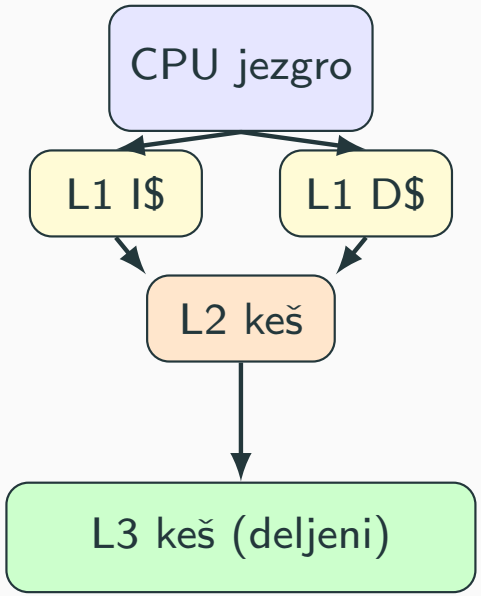


→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: `L1` (najbrži, najmanji), `L2`, `L3`
- Čuvaju **kopije** često korišćenih podataka i instrukcija
- Tipične latencije (približno):

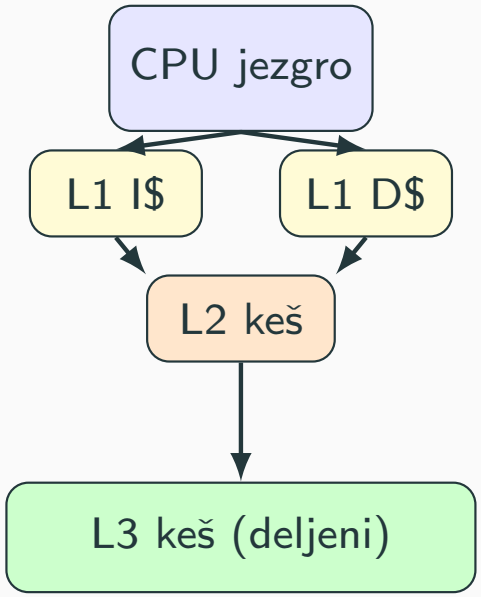


→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: L1 (najbrži, najmanji), L2, L3
- Čuvaju **kopije** često korišćenih podataka i instrukcija
- Tipične latencije (približno):
  - L1: 2–4 ciklusa

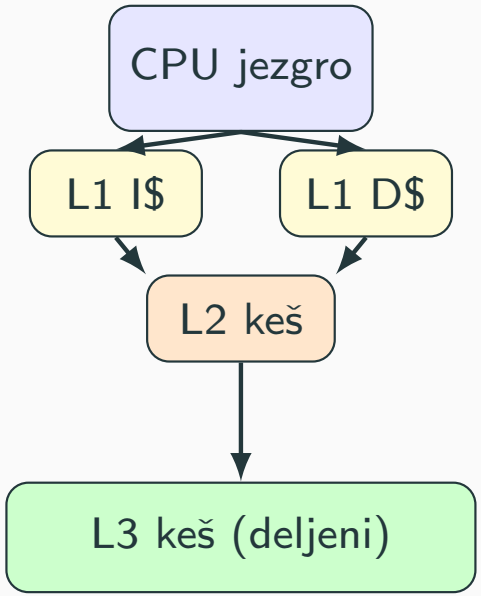


→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: L1 (najbrži, najmanji), L2, L3
- Čuvaju **kopije** često korišćenih podataka i instrukcija
- Tipične latencije (približno):
  - L1: 2–4 ciklusa
  - L2: 10–20 ciklusa

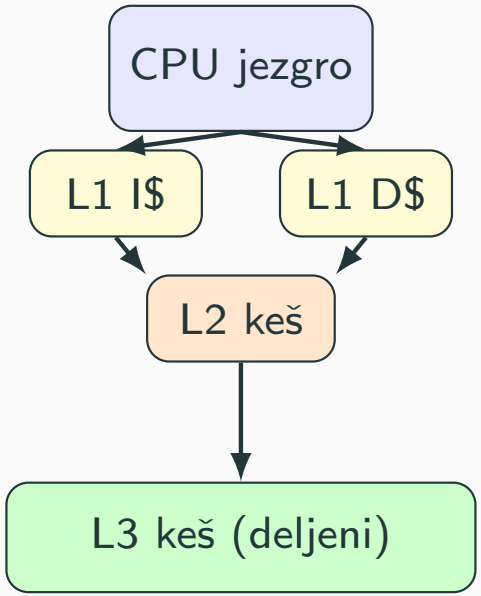


→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Keš memorije: L1, L2, L3

- Keš = mali, brzi međusloj između registara i DRAM-a
- Hijerarhija: L1 (najbrži, najmanji), L2, L3
- Čuvaju **kopije** često korišćenih podataka i instrukcija
- Tipične latencije (približno):
  - L1: 2–4 ciklusa
  - L2: 10–20 ciklusa
  - L3: 30–50 ciklusa



→ DRAM

Keš pokušava da sakrije veliku razliku u brzini između CPU i DRAM-a.

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

- Keš je organizovan u blokove (linije), tipično 32–128 B

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

- Keš je organizovan u blokove (linije), tipično 32–128 B
- Kada CPU traži podatak, dovlači se ceo blok

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

- Keš je organizovan u blokove (linije), tipično 32–128 B
- Kada CPU traži podatak, dovlači se ceo blok
- **Vremenska lokalnost**: isti podatak se uskoro ponovo koristi

Organizacija keša

Osnovni pojmovi

- **Cache hit** – traženi podatak se nalazi u kešu
- **Cache miss** – podatak nije u kešu, mora se dovući iz nižeg nivoa
- Na promašaj, keš bira blok koji će zameniti (LRU, pseudo-LRU, dr.)

- Keš je organizovan u blokove (linije), tipično 32–128 B
- Kada CPU traži podatak, dovlači se ceo blok
- **Vremenska lokalnost**: isti podatak se uskoro ponovo koristi
- **Prostorna lokalnost**: koriste se susedne adrese

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti
- **Write-back** (pisanje-na-izbacivanje)

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti
- **Write-back** (pisanje-na-izbacivanje)
  - Izmene ostaju u kešu, blok se označava kao „prljav“

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti
- **Write-back** (pisanje-na-izbacivanje)
  - Izmene ostaju u kešu, blok se označava kao „prljav“
  - Pri izbacivanju bloka, sadržaj se upisuje u niži nivo

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti
- **Write-back** (pisanje-na-izbacivanje)
  - Izmene ostaju u kešu, blok se označava kao „prljav“
  - Pri izbacivanju bloka, sadržaj se upisuje u niži nivo

Strategije upisa u keš

- **Write-through** (pisanje-prolazom)
  - Svaka izmena u kešu odmah ide i u niži nivo (DRAM)
  - Jednostavnije održavanje konzistentnosti
- **Write-back** (pisanje-na-izbacivanje)
  - Izmene ostaju u kešu, blok se označava kao „prljav“
  - Pri izbacivanju bloka, sadržaj se upisuje u niži nivo

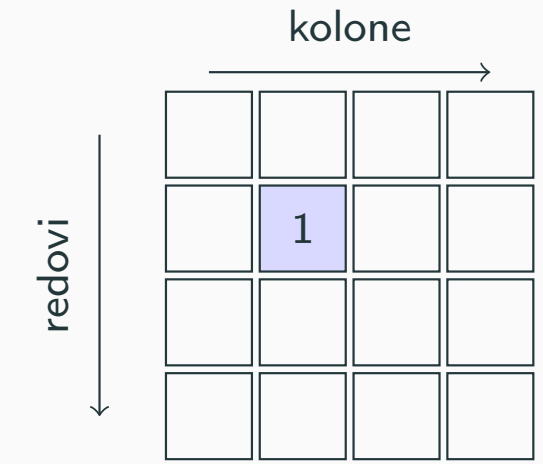
Write-back smanjuje saobraćaj ka memoriji, ali zahteva složeniju kontrolu i može biti kritičan pri kvarovima (podaci nisu još u DRAM-u).

Memorijska hijerarhija

Glavna memorija, SSD i NVM

DRAM – glavna memorija

- **DRAM** – dinamička memorija sa slobodnim pristupom

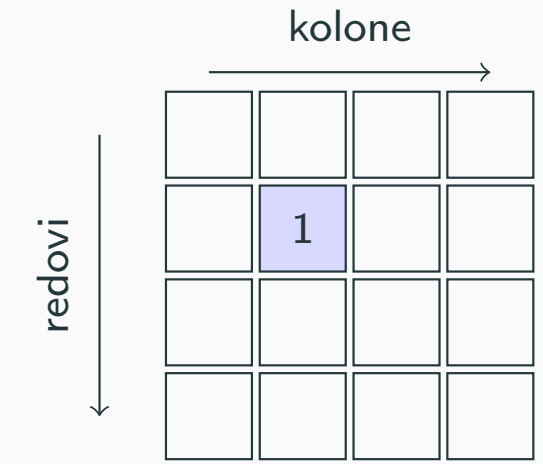


DRAM ćelije organizovane u mrežu
(svaki kvadratić = 1 bit)

Tipično vreme pristupa: 50–100 ns (desetine puta sporije od keša, stotine puta od registara).

DRAM – glavna memorija

- **DRAM** – dinamička memorija sa slobodnim pristupom
- Svaka ćelija: tranzistor + kondenzator

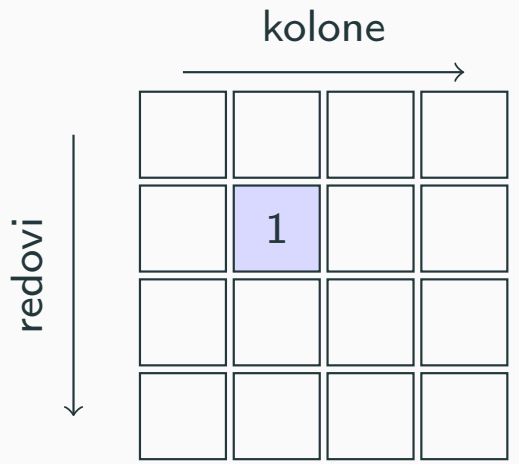


DRAM ćelije organizovane u mrežu
(svaki kvadratić = 1 bit)

Tipično vreme pristupa: 50–100 ns (desetine puta sporije od keša, stotine puta od registara).

DRAM – glavna memorija

- **DRAM** – dinamička memorija sa slobodnim pristupom
- Svaka ćelija: tranzistor + kondenzator
- Napon se „gubi“ vremenom → potrebna periodična obnova (refresh)

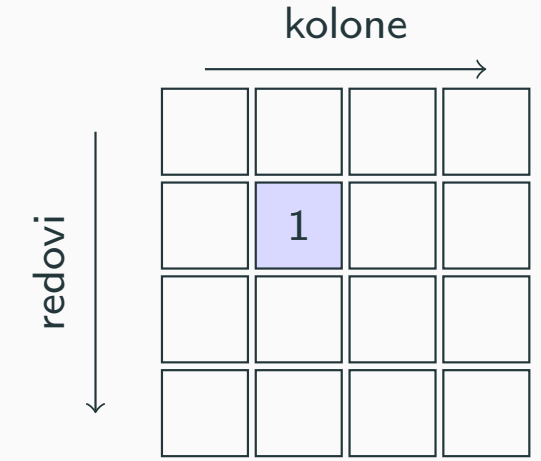


DRAM ćelije organizovane u mrežu
(svaki kvadratić = 1 bit)

Tipično vreme pristupa: 50–100 ns (desetine puta sporije od keša, stotine puta od registara).

DRAM – glavna memorija

- **DRAM** – dinamička memorija sa slobodnim pristupom
- Svaka ćelija: tranzistor + kondenzator
- Napon se „gubi“ vremenom → potrebna periodična obnova (refresh)
- Volatilna – gubi sadržaj pri nestanku napajanja

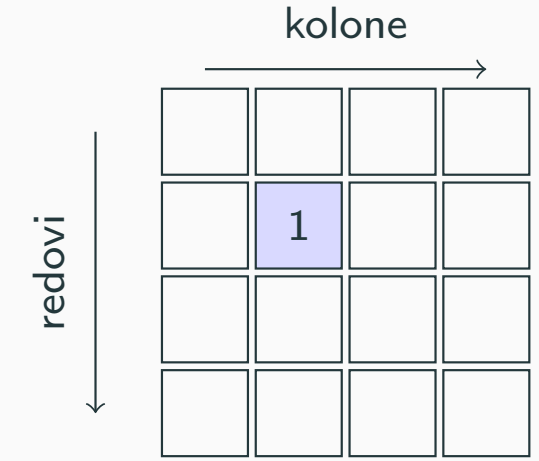


DRAM ćelije organizovane u mrežu
(svaki kvadratić = 1 bit)

Tipično vreme pristupa: 50–100 ns (desetine puta sporije od keša, stotine puta od registara).

DRAM – glavna memorija

- **DRAM** – dinamička memorija sa slobodnim pristupom
- Svaka ćelija: tranzistor + kondenzator
- Napon se „gubi“ vremenom → potrebna periodična obnova (refresh)
- Volatilna – gubi sadržaj pri nestanku napajanja
- Velika gustina, povoljna cena po bitu



DRAM ćelije organizovane u mrežu
(svaki kvadratić = 1 bit)

Tipično vreme pristupa: 50–100 ns (desetine puta sporije od keša, stotine puta od registara).

Uloga DRAM-a u hijerarhiji

Glavna memorija je središnji radni prostor računara:

- Čuva sve programe i podatke koji su trenutno aktivni

Uloga DRAM-a u hijerarhiji

Glavna memorija je središnji radni prostor računara:

- Čuva sve programe i podatke koji su trenutno aktivni
- CPU ne pristupa direktno SSD-u/disku, već se sve pre prvo učitava u DRAM

Uloga DRAM-a u hijerarhiji

Glavna memorija je središnji radni prostor računara:

- Čuva sve programe i podatke koji su trenutno aktivni
- CPU ne pristupa direktno SSD-u/disku, već se sve pre prvo učitava u DRAM
- Što je manje promašaja keša ka DRAM-u, to je niže prosečno vreme pristupa (AMAT) i manja potrošnja energije

Uloga DRAM-a u hijerarhiji

Glavna memorija je središnji radni prostor računara:

- Čuva sve programe i podatke koji su trenutno aktivni
- CPU ne pristupa direktno SSD-u/disku, već se sve pre prvo učitava u DRAM
- Što je manje promašaja keša ka DRAM-u, to je niže prosečno vreme pristupa (AMAT) i manja potrošnja energije

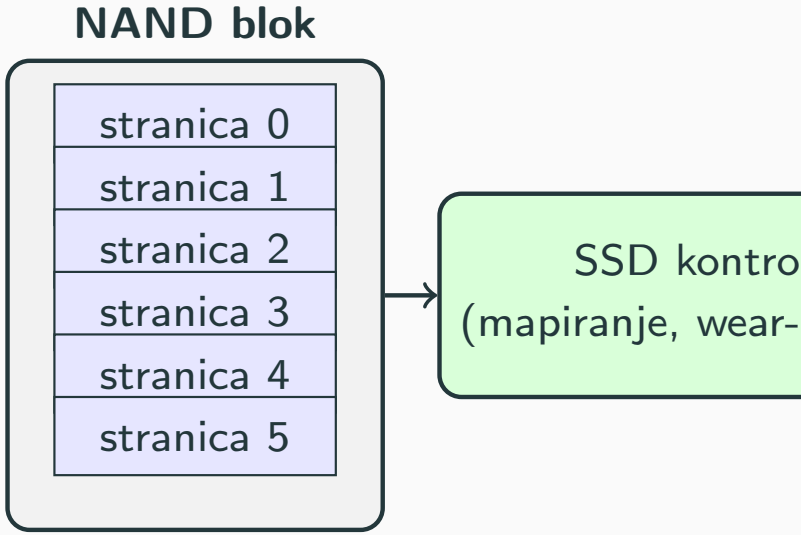
Uloga DRAM-a u hijerarhiji

Glavna memorija je središnji radni prostor računara:

- Čuva sve programe i podatke koji su trenutno aktivni
- CPU ne pristupa direktno SSD-u/disku, već se sve pre prvo učitava u DRAM
- Što je manje promašaja keša ka DRAM-u, to je niže prosečno vreme pristupa (AMAT) i manja potrošnja energije

SSD – nevolatilno skladište

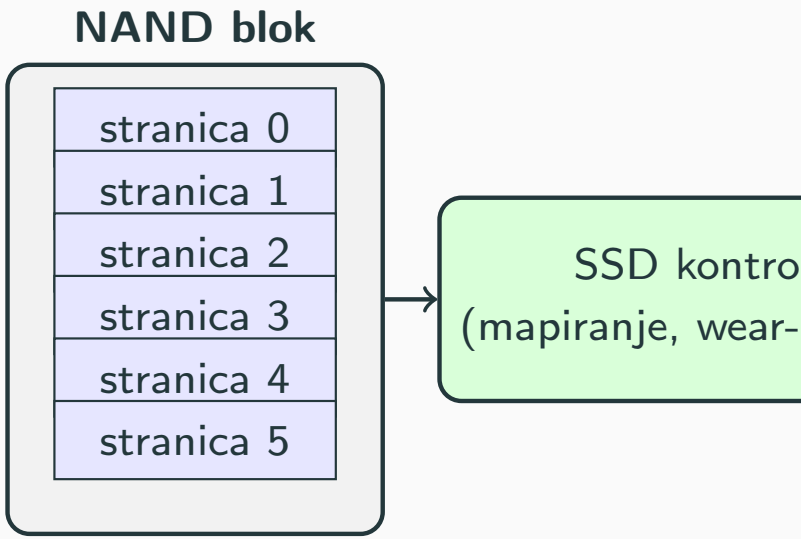
- SSD koristi **NAND tranzistora**



blok se briše ceo pre ponovnog pisanja

SSD – nevolatilno skladište

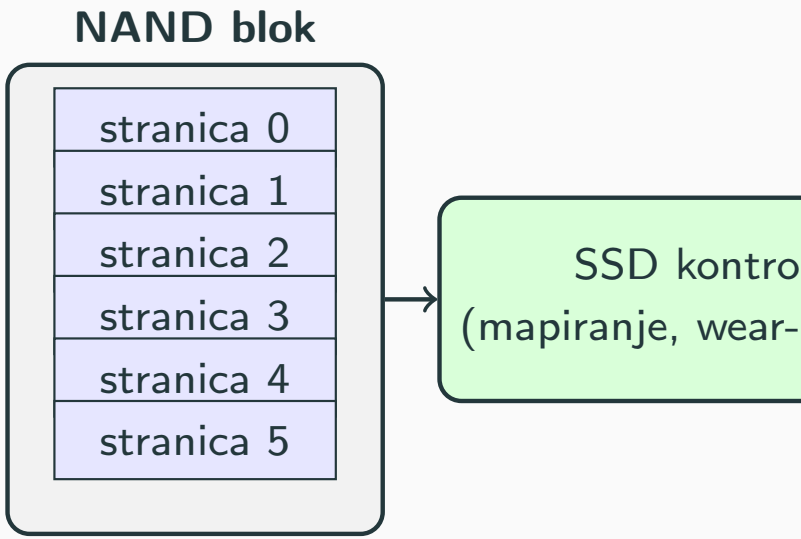
- SSD koristi **NAND tranzistora**
- Nevolatilan – zadržava podatke bez napajanja



blok se briše ceo pre ponovnog pisanja

SSD – nevolatilno skladište

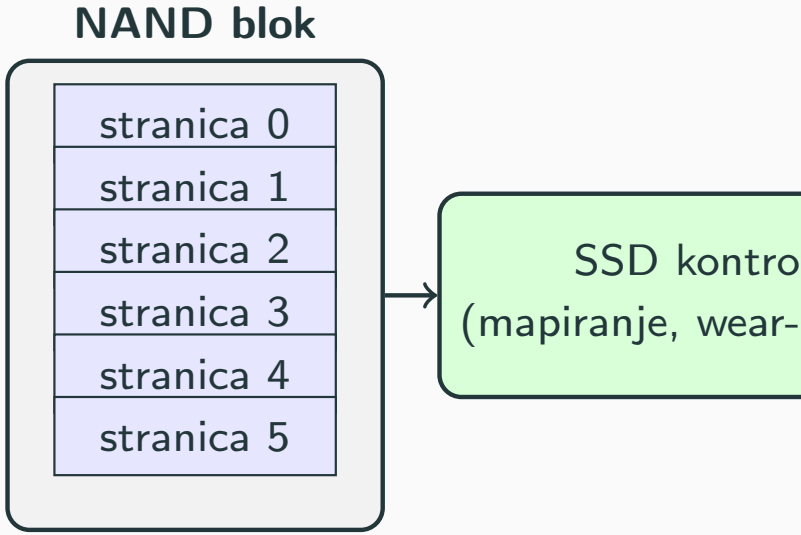
- SSD koristi **NAND tranzistora**
- Nevolatilan – zadržava podatke bez napajanja
- Operiše nad stranicama i blokovima, ne nad bajtovima



blok se briše ceo pre ponovnog pisanja

SSD – nevolatilno skladište

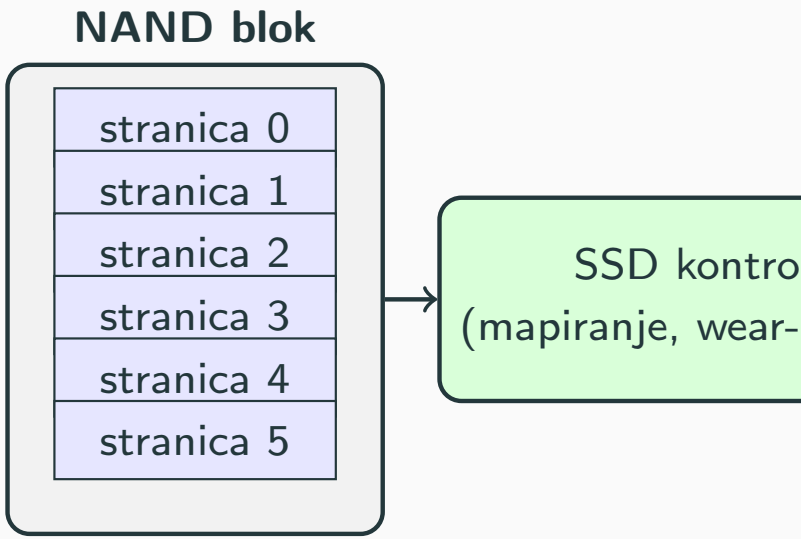
- SSD koristi **NAND tranzistora**
- Nevolatilan – zadržava podatke bez napajanja
- Operiše nad stranicama i blokovima, ne nad bajtovima
- Pre upisa blok se mora obrisati → erase-before-write



blok se briše ceo pre ponovnog pisanja

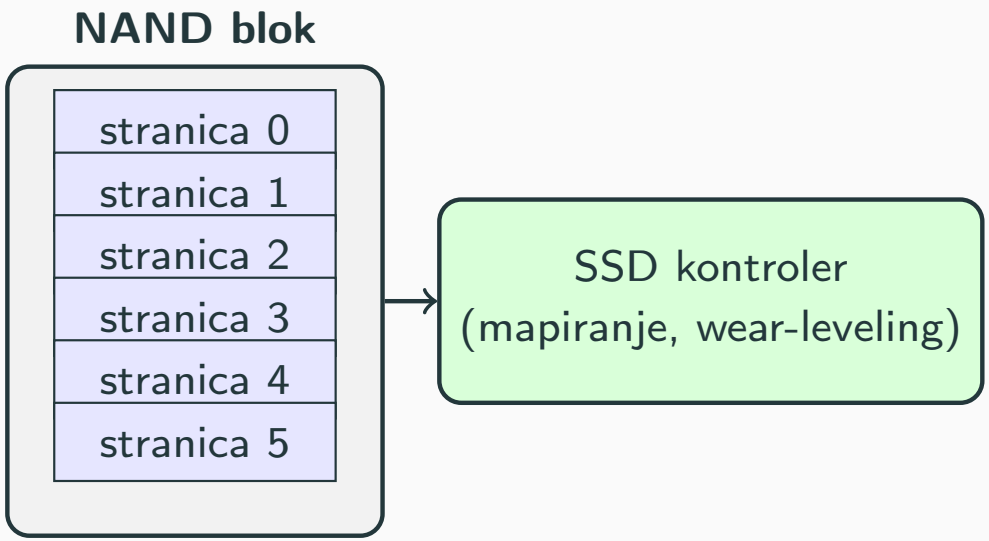
SSD – nevolatilno skladište

- SSD koristi **NAND tranzistora**
- Nevolatilan – zadržava podatke bez napajanja
- Operiše nad stranicama i blokovima, ne nad bajtovima
- Pre upisa blok se mora obrisati → erase-before-write
- Latencija u mikrosekundama (sporije od DRAM-a, brže od HDD)



blok se briše ceo pre ponovnog pisanja

SSD – nevolatilno skladište



blok se briše ceo pre ponovnog pisanja

- Mapiranje logičkih na fizičke adrese
- Keširanje
- Upravljanje ciklusima pisanja da se produži vek trajanja

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost
- Primeri: Intel Optane (3D XPoint), PCM, MRAM

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost
- Primeri: Intel Optane (3D XPoint), PCM, MRAM
- Često podržavaju adresiranje po bajtu (za razliku od NAND)

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost
- Primeri: Intel Optane (3D XPoint), PCM, MRAM
- Često podržavaju adresiranje po bajtu (za razliku od NAND)
- Koriste fizičke ili magnetne promene u materijalu koje ostaju i bez napajanja.

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost
- Primeri: Intel Optane (3D XPoint), PCM, MRAM
- Često podržavaju adresiranje po bajtu (za razliku od NAND)
- Koriste fizičke ili magnetne promene u materijalu koje ostaju i bez napajanja.

NVM – nevolatilne memorije srednjeg sloja

- Popunjavaju jaz između DRAM-a i SSD-a
- Kombinuju: relativno brz pristup i trajnost
- Primeri: Intel Optane (3D XPoint), PCM, MRAM
- Često podržavaju adresiranje po bajtu (za razliku od NAND)
- Koriste fizičke ili magnetne promene u materijalu koje ostaju i bez napajanja.

- Kao **brzi disk** (blokovski uređaj)
- Kao **proširenje radne memorije**
- Kao **trajna radna memorija** – podaci preživljavaju restart

Sistemska magistrala i komunikacija komponenti

Zašto veze između komponenti bitne?

- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.

Zašto veze između komponenti bitne?

- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.
- Ali performanse sistema zavise i od toga kako su komponente povezane.

Zašto veze između komponenti bitne?

- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.
- Ali performanse sistema zavise i od toga kako su komponente povezane.
- CPU, RAM, SSD, GPU, mreža – stalno razmenjuju podatke.

Zašto veze između komponenti bitne?

- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.
- Ali performanse sistema zavise i od toga kako su komponente povezane.
- CPU, RAM, SSD, GPU, mreža – stalno razmenjuju podatke.
- Ako je veza spora ili zagušena $\rightarrow$ sistem čeka, bez obzira na brz CPU.

Zašto veze između komponenti bitne?

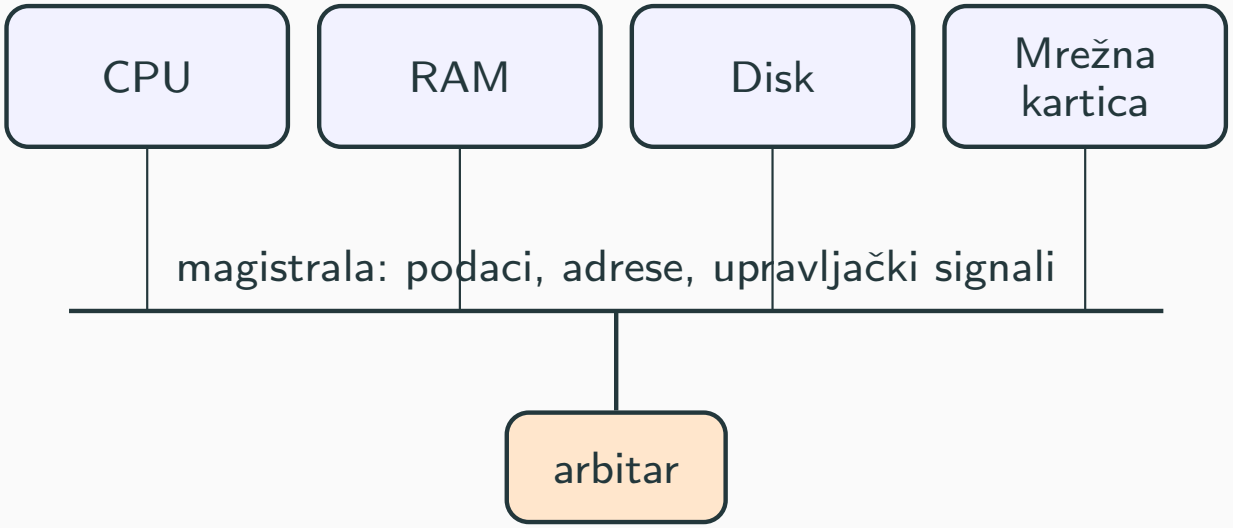
- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.
- Ali performanse sistema zavise i od toga kako su komponente povezane.
- CPU, RAM, SSD, GPU, mreža – stalno razmenjuju podatke.
- Ako je veza spora ili zagušena $\rightarrow$ sistem čeka, bez obzira na brz CPU.

Zašto veze između komponenti bitne?

- Do sada: fokus na **CPU** i **memorijsku hijerarhiju**.
- Ali performanse sistema zavise i od toga kako su komponente povezane.
- CPU, RAM, SSD, GPU, mreža – stalno razmenjuju podatke.
- Ako je veza spora ili zagušena $\rightarrow$ sistem čeka, bez obzira na brz CPU.

Arhitektura magistrala i mreža međupovezivanja je podjednako važna kao i brzina samog procesora.

Klasična sistemska magistrala

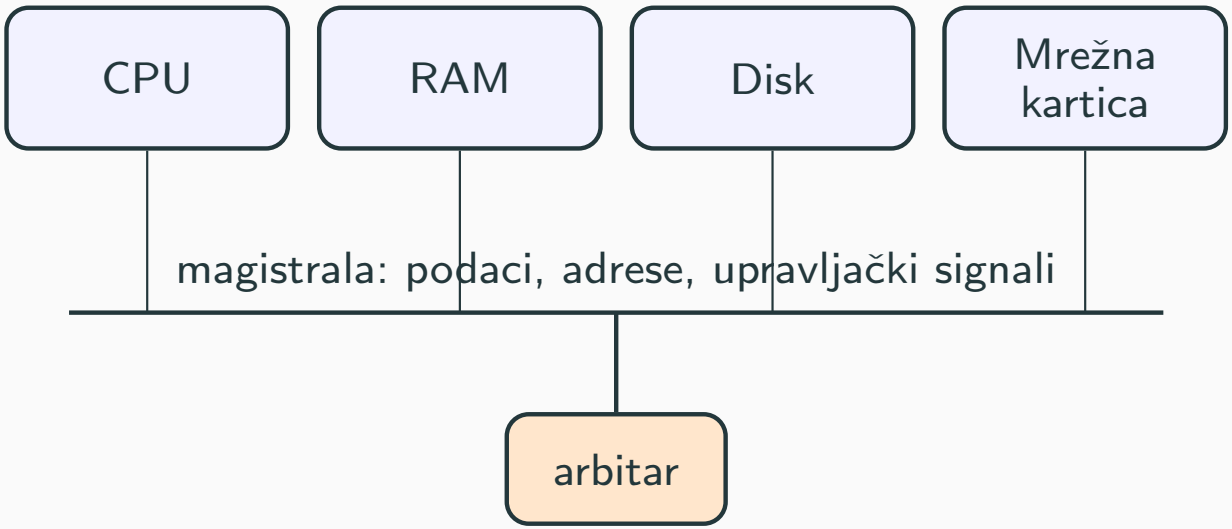


odlučuje koji uređaj trenutno koristi magistralu

Klasična PCI/AGP magistrala: svi uređaji čekaju svoj red na istom kanalu.

Klasična sistemska magistrala

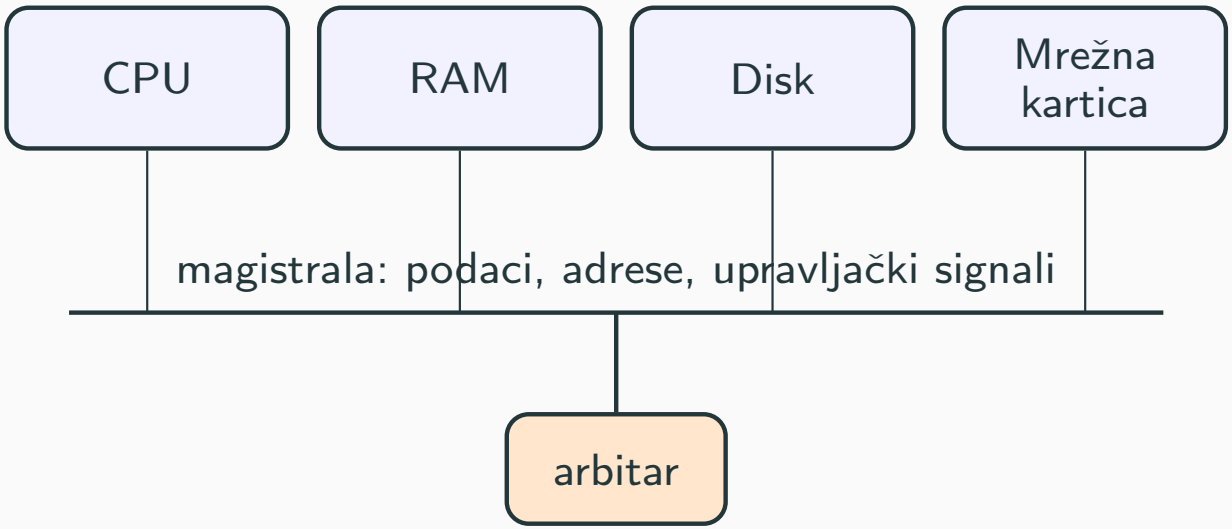
- **Magistrala** = zajednički komunikacioni kanal:



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

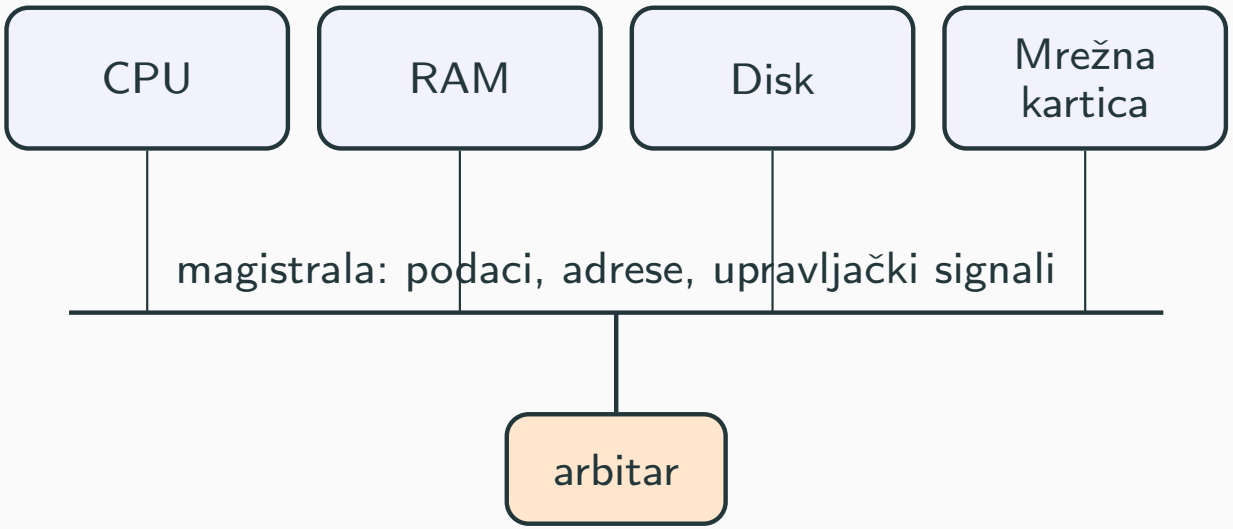
- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

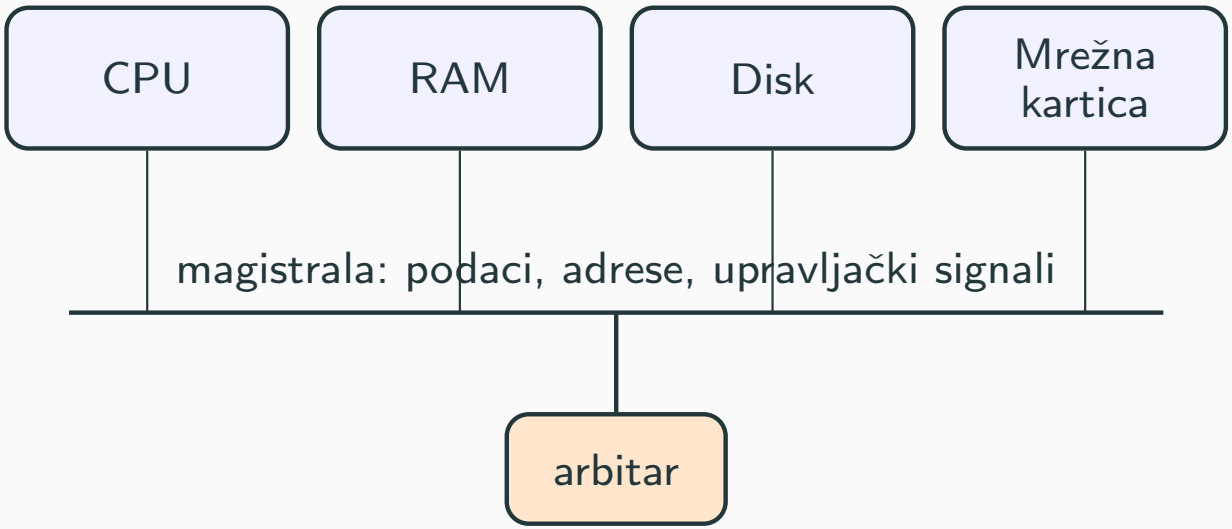
- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke
  - linije za adrese



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

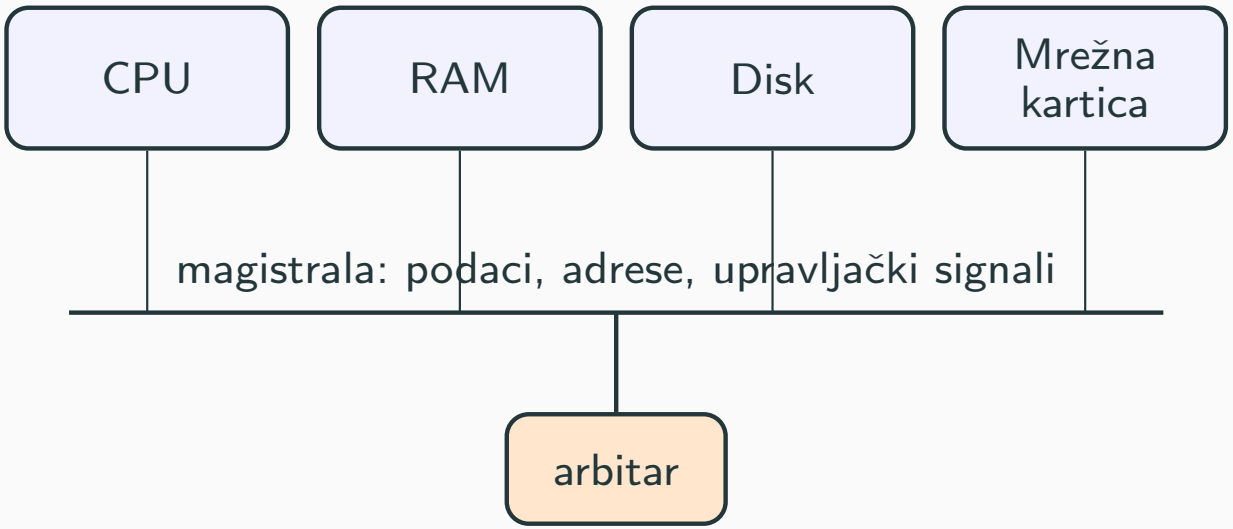
- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke
  - linije za adrese
  - upravljački signali



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

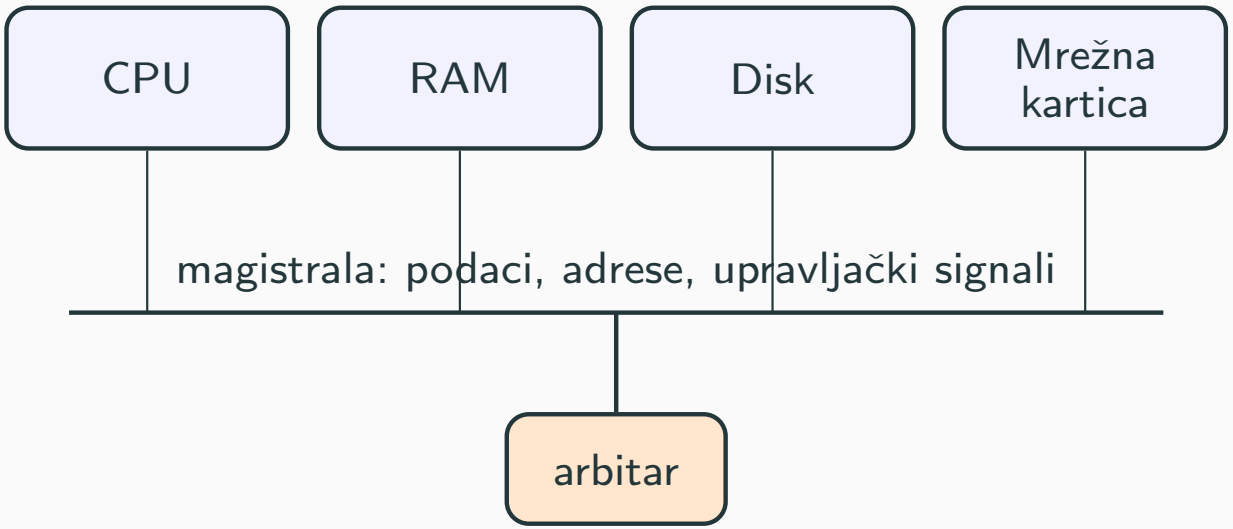
- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke
  - linije za adrese
  - upravljački signali
- Više uređaja deli *isti* kanal.



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

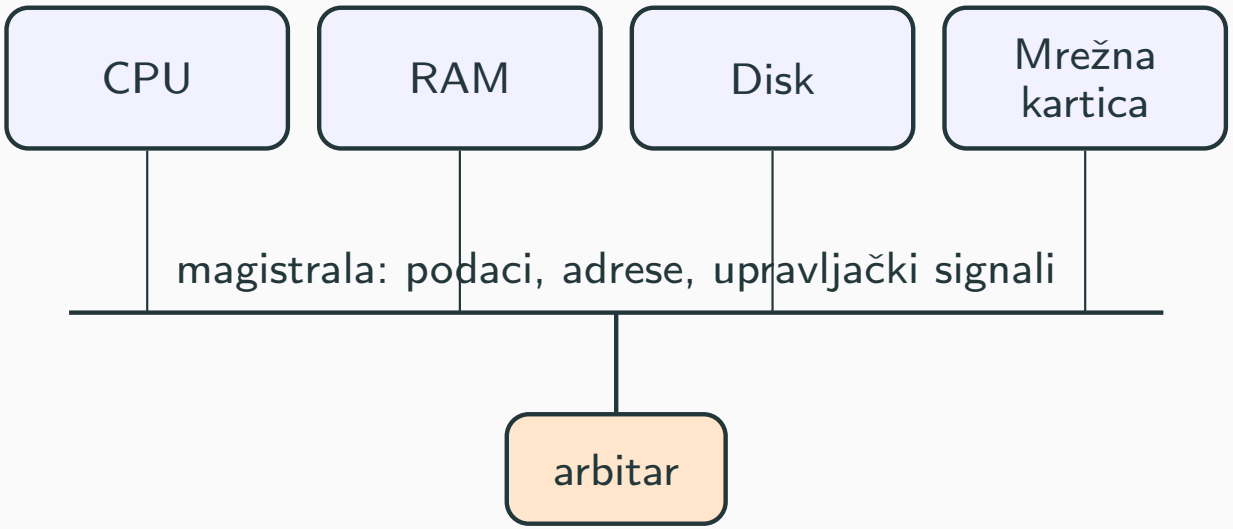
- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke
  - linije za adrese
  - upravljački signali
- Više uređaja deli *isti* kanal.
- Potrebna je arbitraža – ko sada sme da „priča“?



odlučuje koji uređaj trenutno koristi magistralu

Klasična sistemska magistrala

- **Magistrala** = zajednički komunikacioni kanal:
  - linije za podatke
  - linije za adrese
  - upravljački signali
- Više uređaja deli *isti* kanal.
- Potrebna je arbitraža – ko sada sme da „priča“?
- Jednostavno i jeftino rešenje za male sisteme.



odlučuje koji uređaj trenutno koristi magistralu

Propusnost i latencija

- **Propusnost (bandwidth)** – koliko podataka može da prođe u jedinici vremena.

Propusnost i latencija

- **Propusnost (bandwidth)** – koliko podataka može da prođe u jedinici vremena.
- Širina magistrale w (u bitovima) i takt f (u Hz) određuju idealnu propusnost:

$$B = \frac{w \cdot f}{8} \quad [\text{B/s}]$$

Propusnost i latencija

- **Propusnost (bandwidth)** – koliko podataka može da prođe u jedinici vremena.
- Širina magistrale w (u bitovima) i takt f (u Hz) određuju idealnu propusnost:

$$B = \frac{w \cdot f}{8} \quad [\text{B/s}]$$

- **Latencija (latency)** – koliko vremena treba da jedan konkretan podatak stigne do odredišta.

Propusnost i latencija

- **Propusnost (bandwidth)** – koliko podataka može da prođe u jedinici vremena.
- Širina magistrale w (u bitovima) i takt f (u Hz) određuju idealnu propusnost:

$$B = \frac{w \cdot f}{8} \quad [\text{B/s}]$$

- **Latencija (latency)** – koliko vremena treba da jedan konkretan podatak stigne do odredišta.

Propusnost i latencija

- **Propusnost (bandwidth)** – koliko podataka može da prođe u jedinici vremena.
- Širina magistrale w (u bitovima) i takt f (u Hz) određuju idealnu propusnost:

$$B = \frac{w \cdot f}{8} \quad [\text{B/s}]$$

- **Latencija (latency)** – koliko vremena treba da jedan konkretan podatak stigne do odredišta.

64-bitna magistrala na 100 MHz:

$$B = \frac{64 \cdot 100 \cdot 10^6}{8} \approx 0{,}8 \text{ GB/s}$$

Zašto zajednička magistrala postaje usko grlo?

- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.

Zašto zajednička magistrala postaje usko grlo?

- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.
- **Arbitraža** mora da raspodeli pristup $\rightarrow$ svi čekaju.

Zašto zajednička magistrala postaje usko grlo?

- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.
- **Arbitraža** mora da raspodeli pristup $\rightarrow$ svi čekaju.
- Efektivna propusnost po uređaju opada, latencija raste.

Zašto zajednička magistrala postaje usko grlo?

- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.
- **Arbitraža** mora da raspodeli pristup $\rightarrow$ svi čekaju.
- Efektivna propusnost po uređaju opada, latencija raste.
- Rezultat: sistem čeka na prenos, iako CPU i memorija mogu brže.

Zašto zajednička magistrala postaje usko grlo?

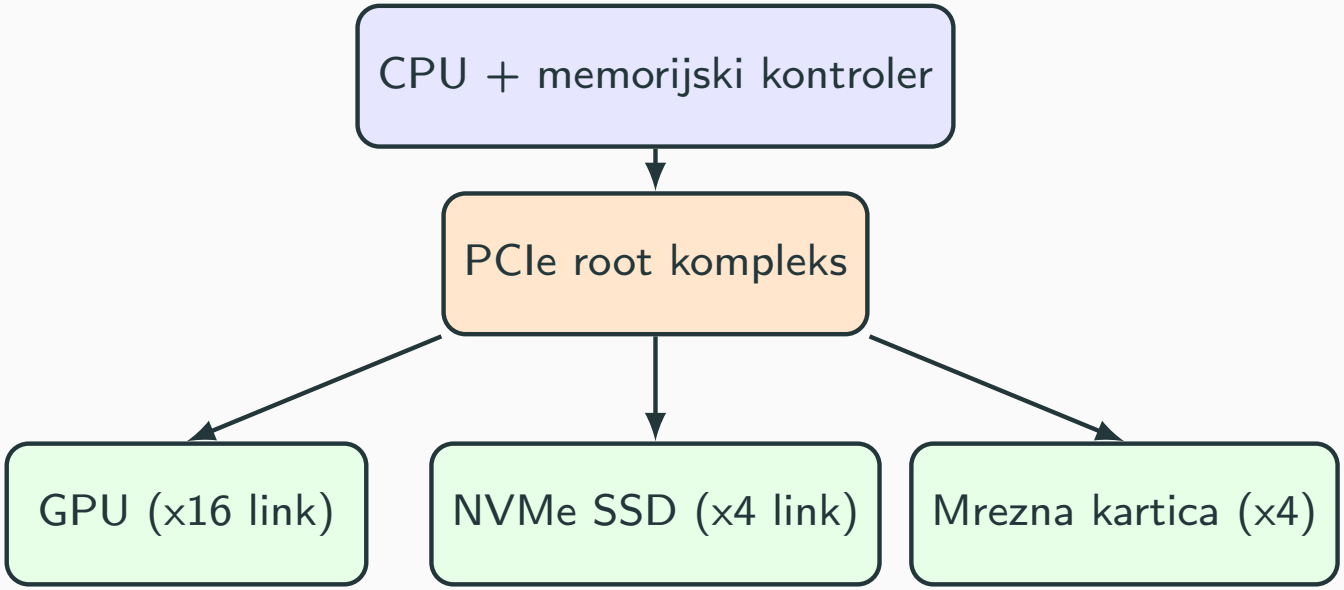
- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.
- **Arbitraža** mora da raspodeli pristup $\rightarrow$ svi čekaju.
- Efektivna propusnost po uređaju opada, latencija raste.
- Rezultat: sistem čeka na prenos, iako CPU i memorija mogu brže.

Zašto zajednička magistrala postaje usko grlo?

- Kako raste broj uređaja (GPU, NVMe, mreža 10/40GbE, USB, ...), više njih želi pristup istom kanalu.
- **Arbitraža** mora da raspodeli pristup $\rightarrow$ svi čekaju.
- Efektivna propusnost po uređaju opada, latencija raste.
- Rezultat: sistem čeka na prenos, iako CPU i memorija mogu brže.

Umesto jedne zajedničke magistrale $\rightarrow$ **mreža međupovezivanja** sa više paralelnih veza (point-to-point).

Od sistemske magistrale do PCI Express (PCIe)

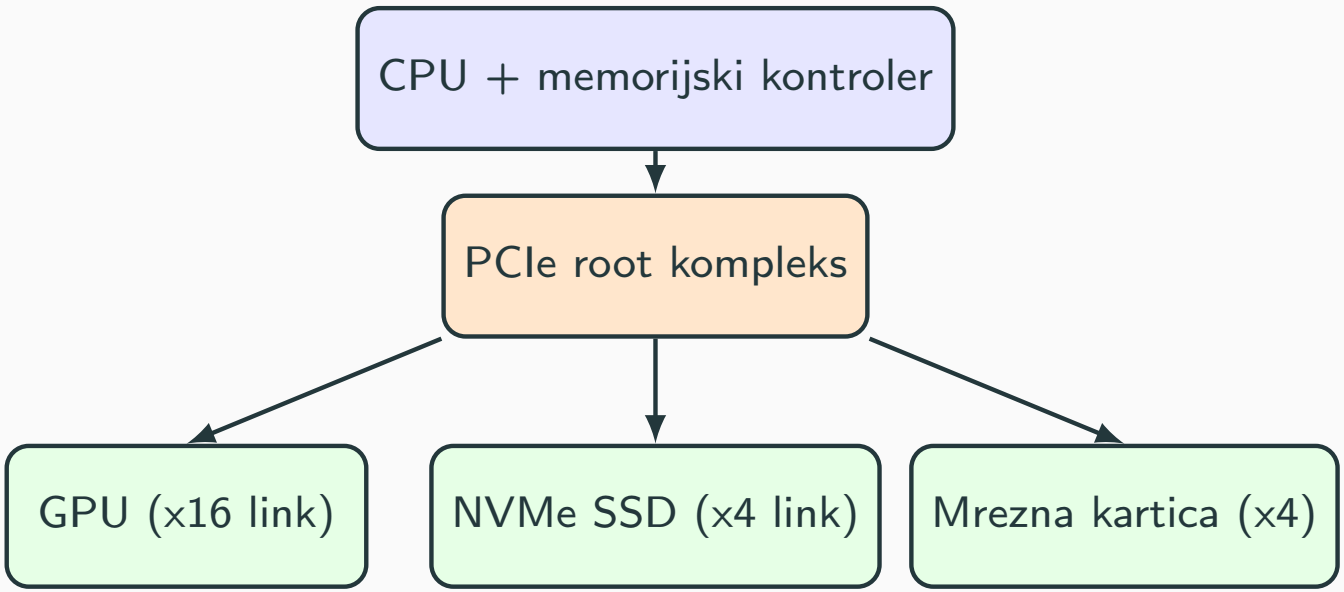


Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

PCIe uklanja usko grlo deljene magistrale i omogućava visok protok ka GPU, NVMe i drugim brzim uređajima.

Od sistemske magistrale do PCI Express (PCIe)

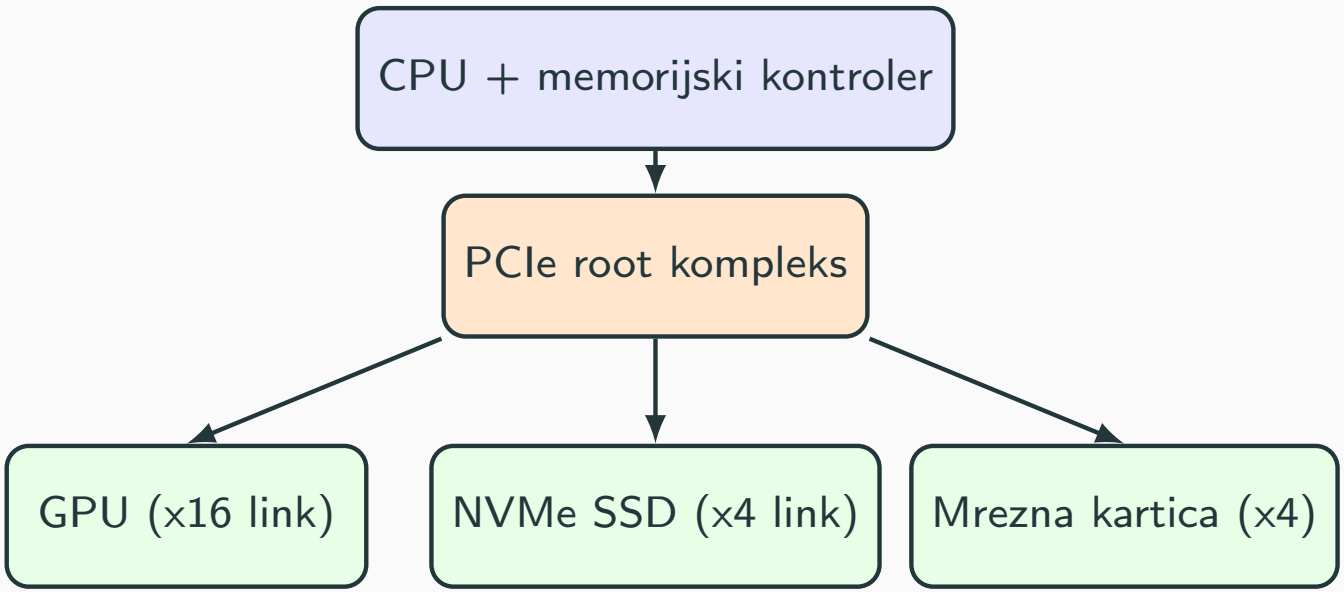
- Stari standardi: **PCI**, **AGP** – deljena magistrala.



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Od sistemske magistrale do PCI Express (PCIe)

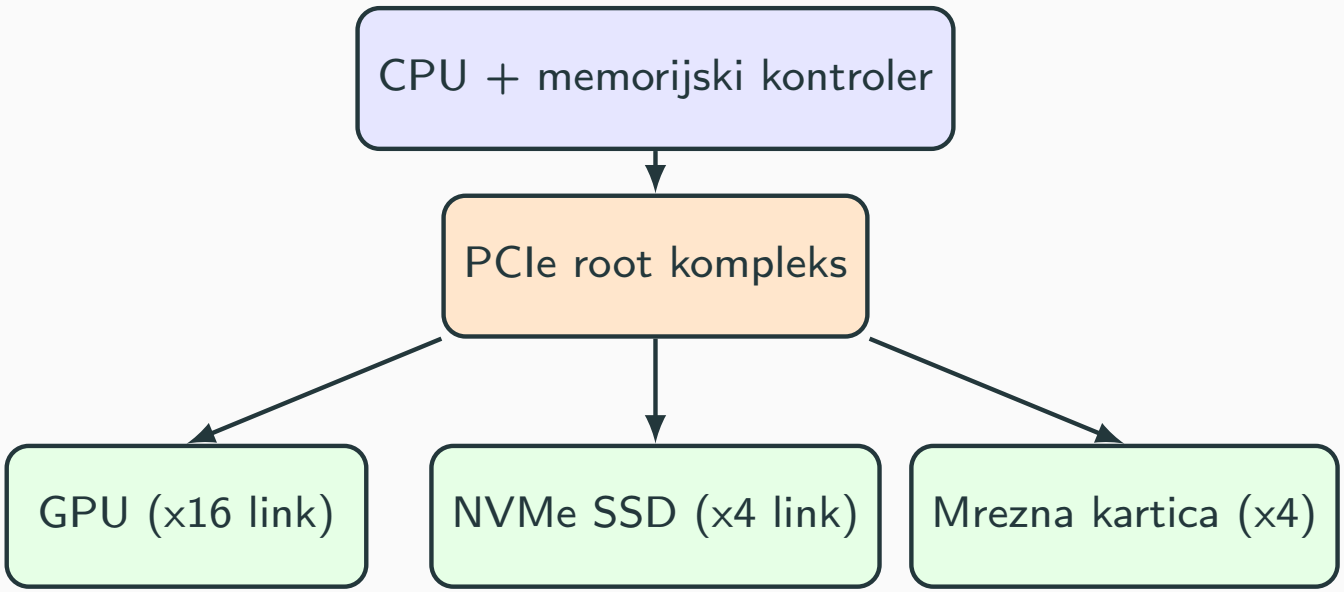
- Stari standardi: **PCI**, **AGP** – deljena magistrala.
- Naslednik: **PCI Express (PCIe)**:



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Od sistemske magistrale do PCI Express (PCIe)

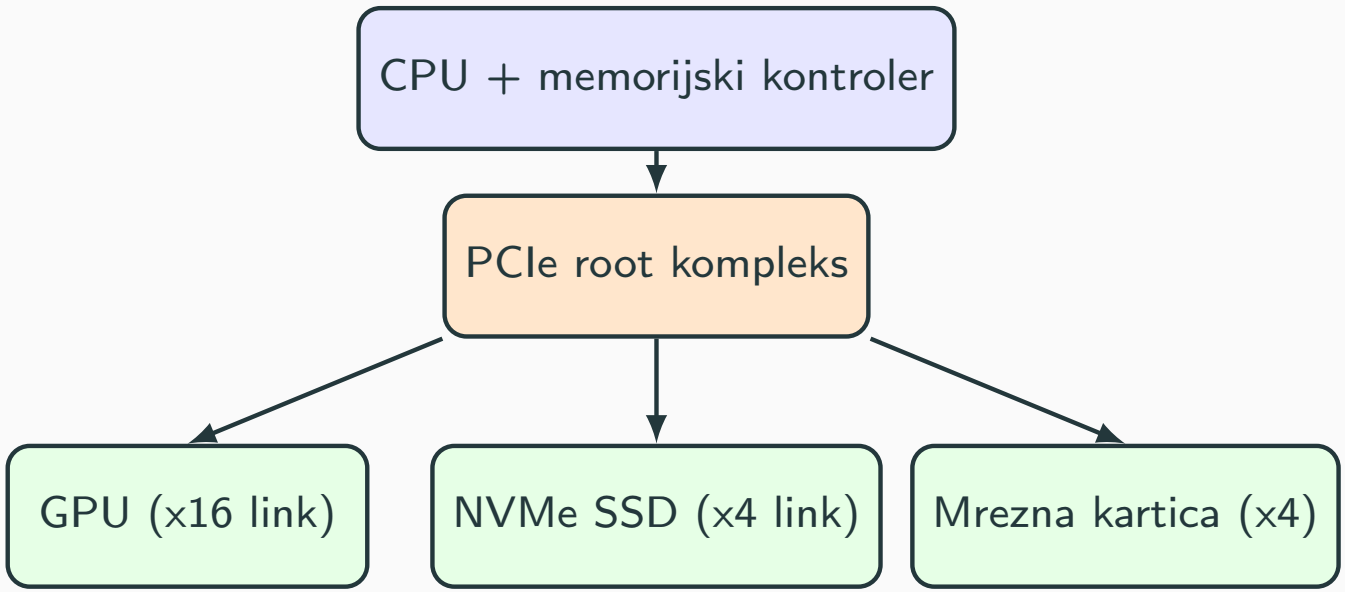
- Stari standardi: **PCI**, **AGP** – deljena magistrala.
- Naslednik: **PCI Express (PCIe)**:
  - point-to-point digitalne veze



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Od sistemske magistrale do PCI Express (PCIe)

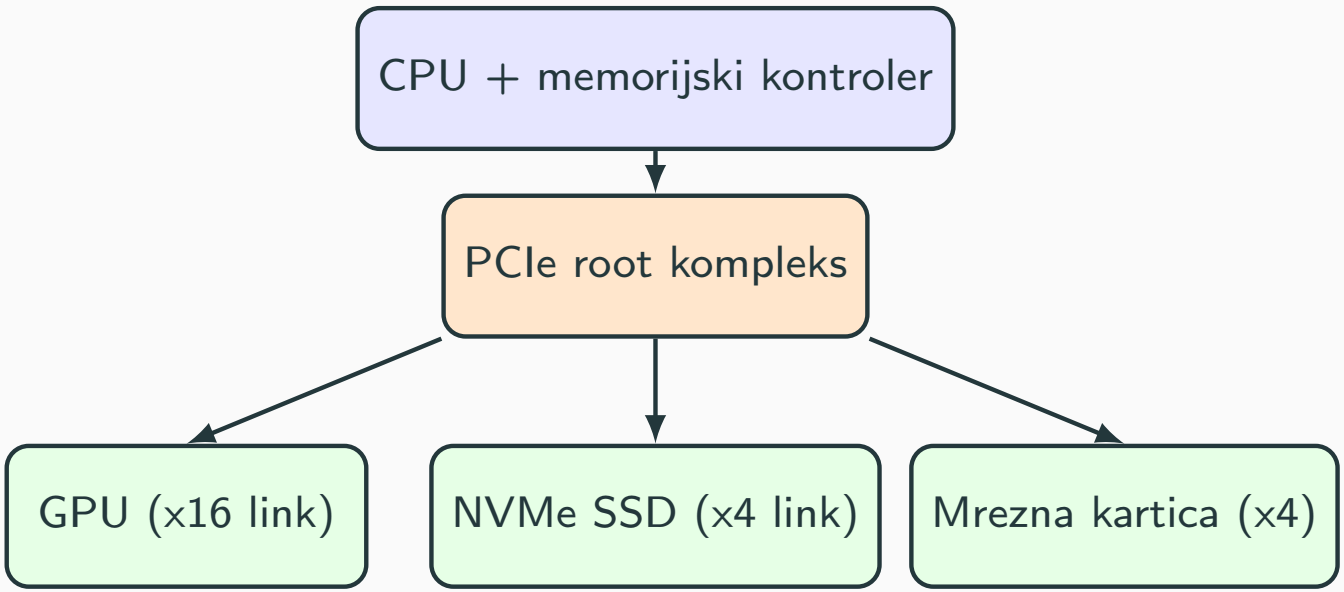
- Stari standardi: **PCI**, **AGP** – deljena magistrala.
- Naslednik: **PCI Express (PCIe)**:
  - point-to-point digitalne veze
  - svaka komponenta ima svoj kanal ka root-kompleksu/čipsetu



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Od sistemske magistrale do PCI Express (PCIe)

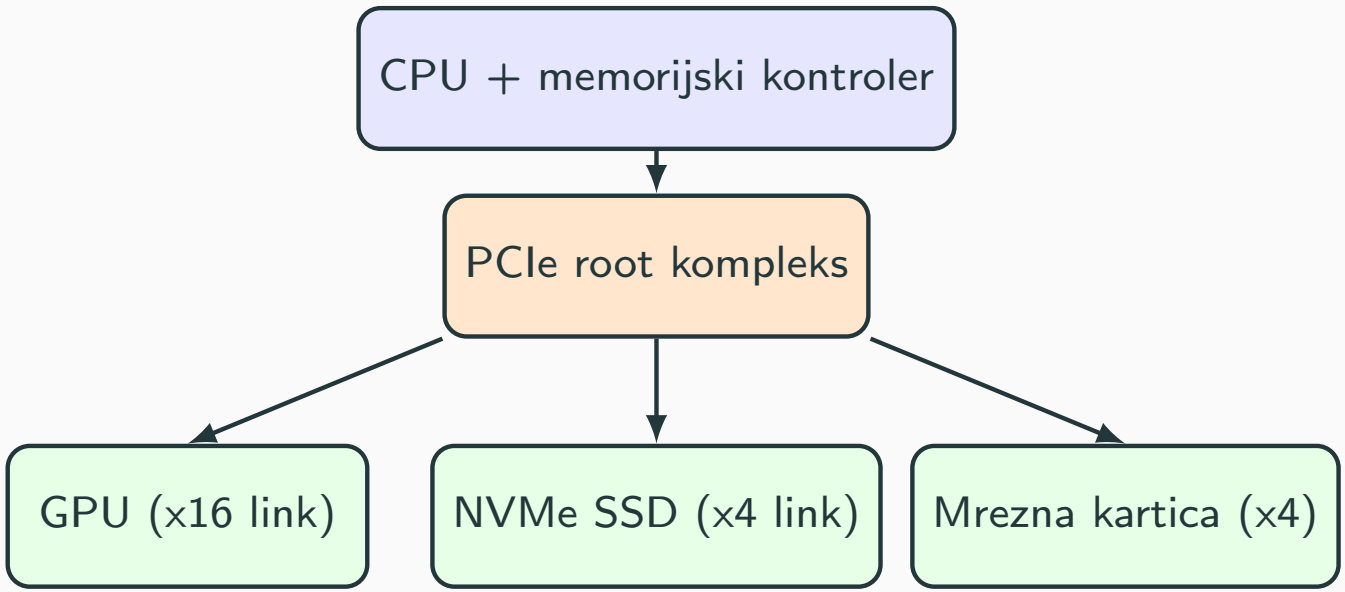
- Stari standardi: **PCI**, **AGP** – deljena magistrala.
- Naslednik: **PCI Express (PCIe)**:
  - point-to-point digitalne veze
  - svaka komponenta ima svoj kanal ka root-kompleksu/čipsetu
  - više prenosa može teći istovremeno



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Od sistemske magistrale do PCI Express (PCIe)

- Stari standardi: **PCI**, **AGP** – deljena magistrala.
- Naslednik: **PCI Express (PCIe)**:
  - point-to-point digitalne veze
  - svaka komponenta ima svoj kanal ka root-kompleksu/čipsetu
  - više prenosa može teći istovremeno
- PCIe *linije* (x1, x4, x8, x16) – više linija = veća propusnost.



Svaki uredjaj ima sopstveni point-to-point PCIe link ka root-kompleksu.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.
  - DMA sam prenosi podatke između uređaja i memorije.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.
  - DMA sam prenosi podatke između uređaja i memorije.
  - CPU se „oslobađa“ i može da radi nešto drugo.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.
  - DMA sam prenosi podatke između uređaja i memorije.
  - CPU se „oslobađa“ i može da radi nešto drugo.
- PCIe + DMA = efikasan, paralelan prenos velikih količina podataka.

Direktan pristup memoriji (DMA) preko PCIe

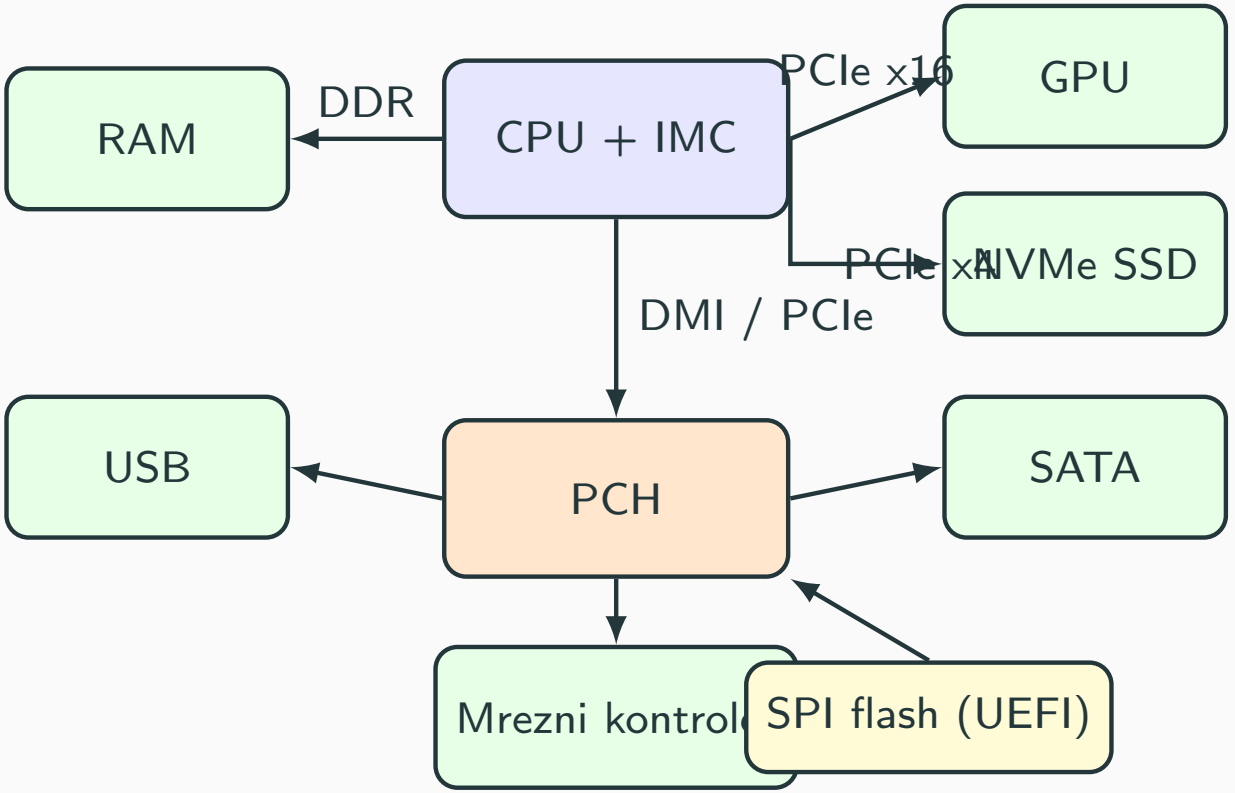
- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.
  - DMA sam prenosi podatke između uređaja i memorije.
  - CPU se „oslobađa“ i može da radi nešto drugo.
- PCIe + DMA = efikasan, paralelan prenos velikih količina podataka.

Direktan pristup memoriji (DMA) preko PCIe

- Klasičan I/O: CPU čita podatke sa uređaja bajt po bajt i piše u RAM.
- Sa **DMA** kontrolerom:
  - CPU zada komandu: „kopiraj blok podataka odavde do ovde“.
  - DMA sam prenosi podatke između uređaja i memorije.
  - CPU se „oslobađa“ i može da radi nešto drugo.
- PCIe + DMA = efikasan, paralelan prenos velikih količina podataka.

Kada DMA završi prenos, generiše prekid – procesor dobija signal da su podaci spremni.

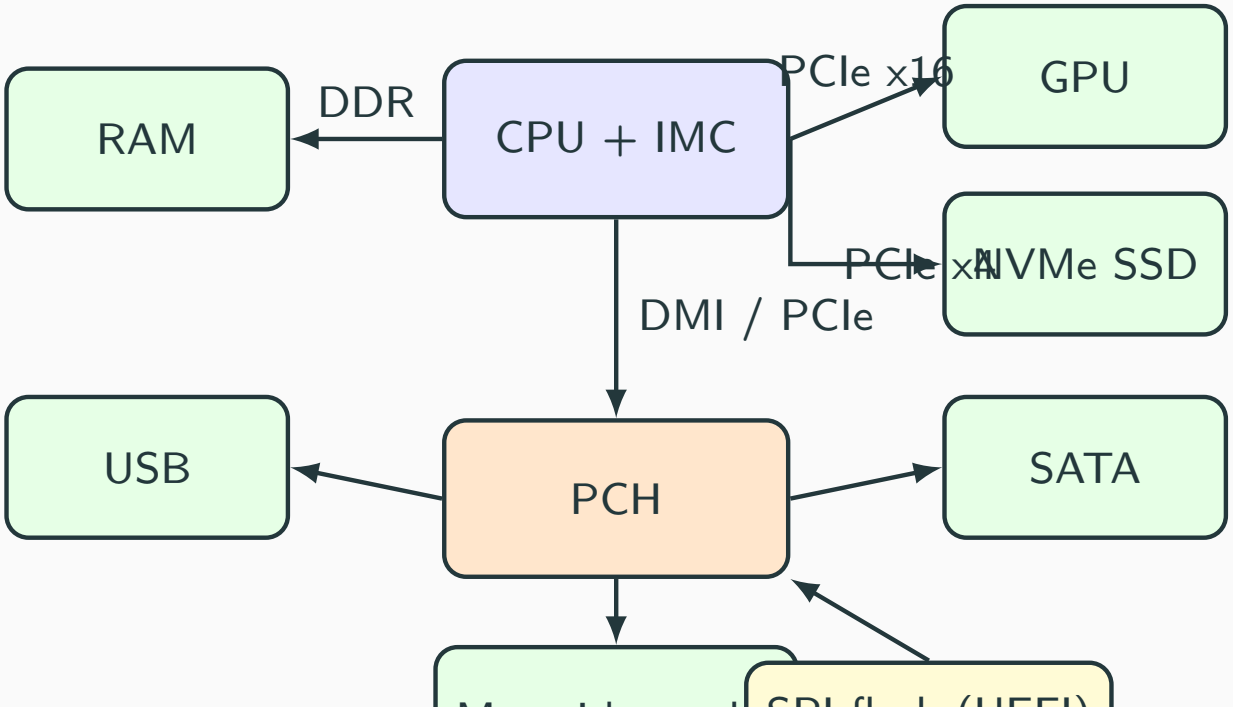
Čipset i specijalizovani kontroleri



Čipset + kontroleri = „saobraćajna policija“ koja usmerava protok podataka između CPU, memorije i uređaja.

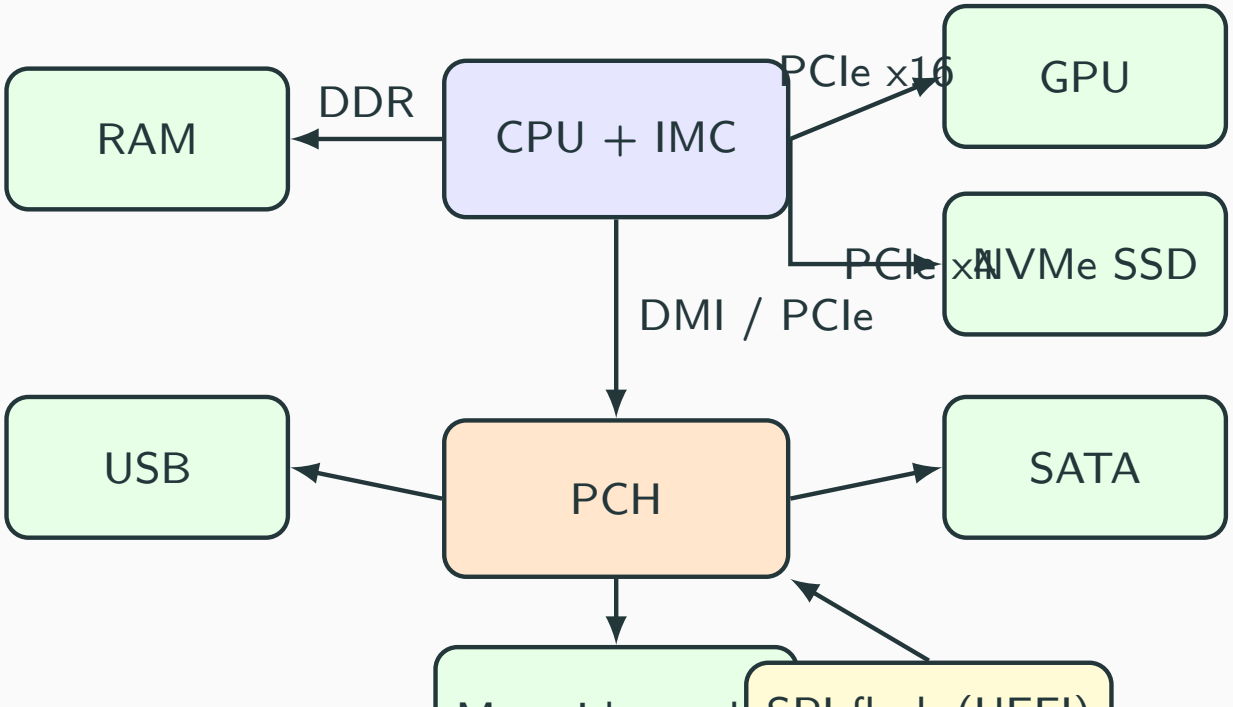
Čipset i specijalizovani kontroleri

- Nekada: **severni most** (CPU–RAM–GPU) i **južni most** (sporiji I/O).



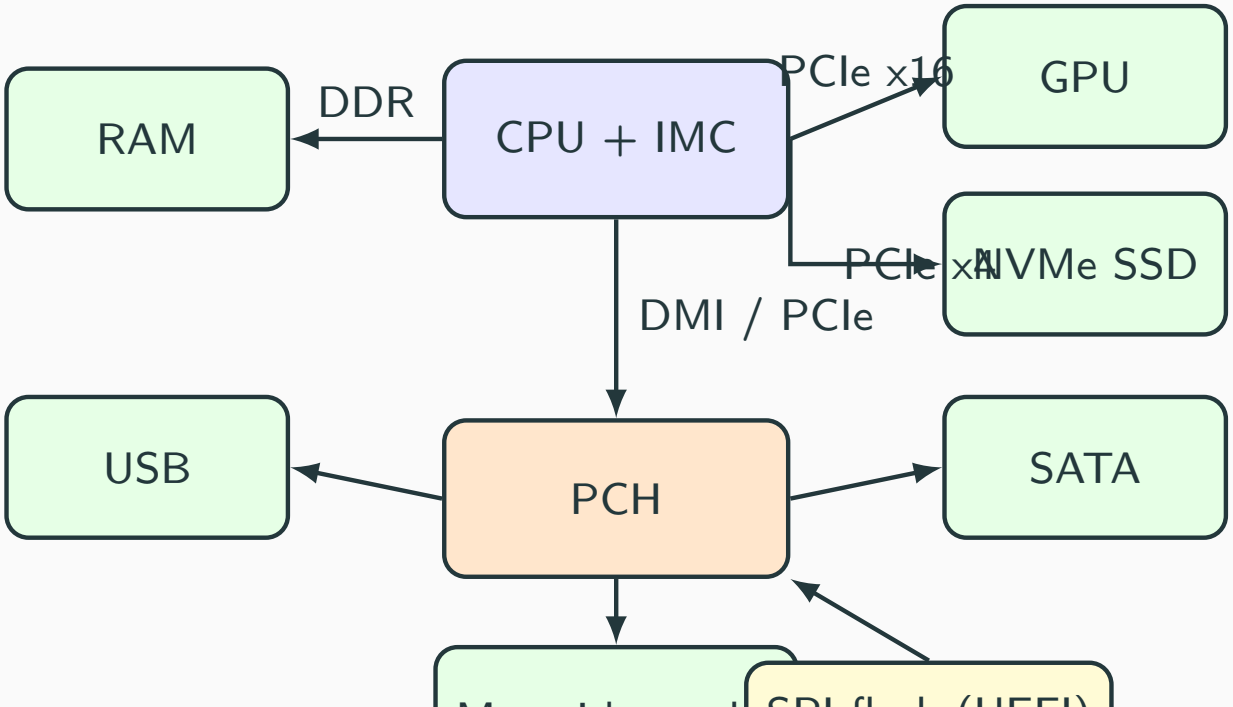
Čipset i specijalizovani kontroleri

- Nekada: **severni most** (CPU–RAM–GPU) i **južni most** (sporiji I/O).
- Danas:



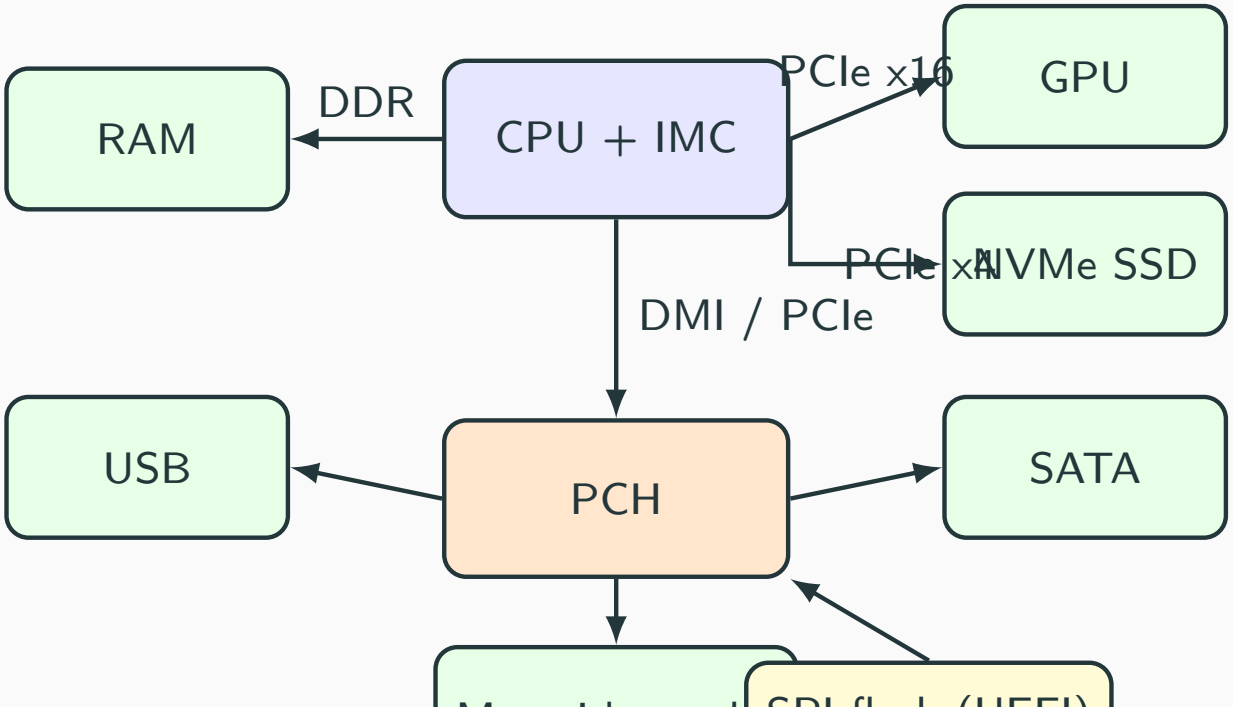
Čipset i specijalizovani kontroleri

- Nekada: **severni most** (CPU–RAM–GPU) i **južni most** (sporiji I/O).
- Danas:
  - memorijski kontroler i PCIe često u samom CPU-u,



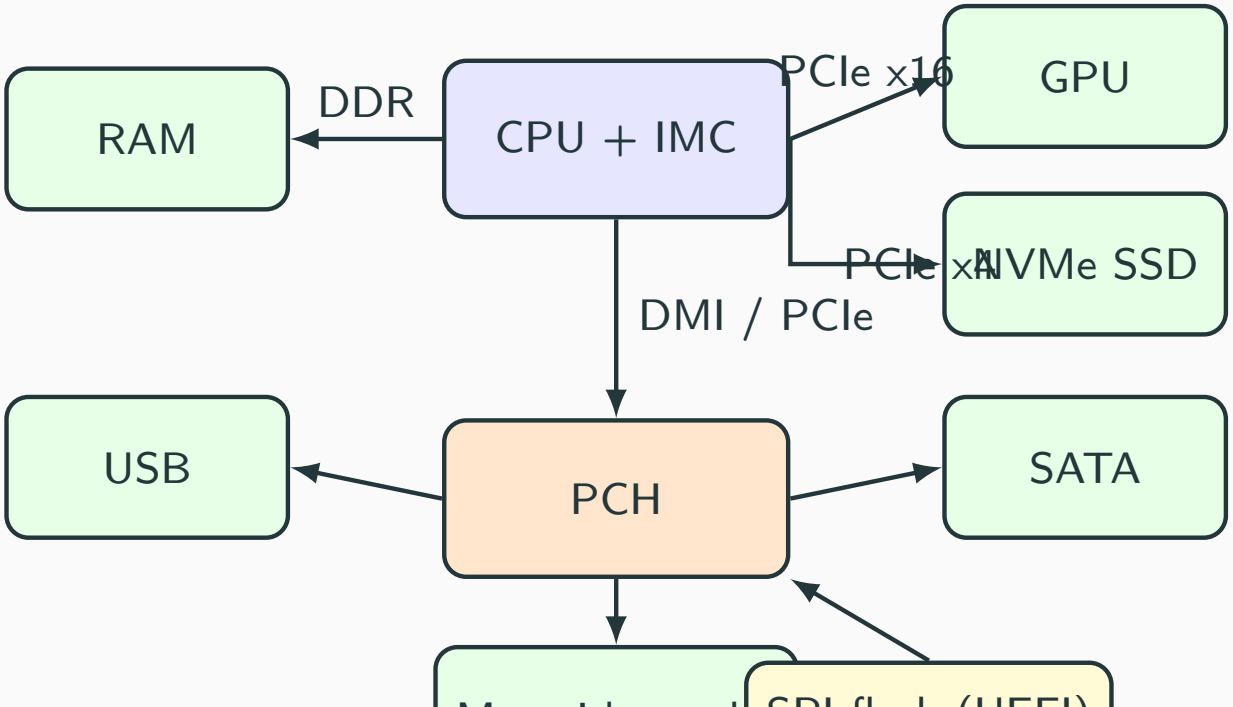
Čipset i specijalizovani kontroleri

- Nekada: **severni most** (CPU–RAM–GPU) i **južni most** (sporiji I/O).
- Danas:
  - memorijski kontroler i PCIe često u samom CPU-u,
  - čipset (PCH) povezuje USB, SATA, dodatne PCIe linije, mrežu...



Čipset i specijalizovani kontroleri

- Nekada: **severni most** (CPU–RAM–GPU) i **južni most** (sporiji I/O).
- Danas:
  - memorijski kontroler i PCIe često u samom CPU-u,
  - čipset (PCH) povezuje USB, SATA, dodatne PCIe linije, mrežu...
- Svaki uređaj ima sopstveni **kontroler** (disk, mreža, USB...).



Ulazno–izlazni podsistem

Ulazno–izlazni podsistem (I/O)

- Uređaji: diskovi, tastature, miševi, mrežne kartice, GPU, USB uređaji...

Ulazno–izlazni podsistem (I/O)

- Uređaji: diskovi, tastature, miševi, mrežne kartice, GPU, USB uređaji...
- Interfejsi i standardi: USB, SATA, PCIe, NVMe, Ethernet...

Ulazno–izlazni podsistem (I/O)

- Uređaji: diskovi, tastature, miševi, mrežne kartice, GPU, USB uređaji...
- Interfejsi i standardi: USB, SATA, PCIe, NVMe, Ethernet...
- Kontroleri: disk kontroler, USB kontroler, mrežni kontroler...

Ulazno–izlazni podsistem (I/O)

- Uređaji: diskovi, tastature, miševi, mrežne kartice, GPU, USB uređaji...
- Interfejsi i standardi: USB, SATA, PCIe, NVMe, Ethernet...
- Kontroleri: disk kontroler, USB kontroler, mrežni kontroler...

Ulazno–izlazni podsistem (I/O)

- Uređaji: diskovi, tastature, miševi, mrežne kartice, GPU, USB uređaji...
- Interfejsi i standardi: USB, SATA, PCIe, NVMe, Ethernet...
- Kontroleri: disk kontroler, USB kontroler, mrežni kontroler...

I/O podsistem je sloj koji „spaja“ procesor i memoriju sa spoljnim svetom – bez njega računar bi bio zatvorena kutija.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).
  - Kontroler prevodi komandu u električne signale i protokol uređaja.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).
  - Kontroler prevodi komandu u električne signale i protokol uređaja.
  - Kada je gotovo, javlja CPU-u preko prekida.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).
  - Kontroler prevodi komandu u električne signale i protokol uređaja.
  - Kada je gotovo, javlja CPU-u preko prekida.
- Prednost: CPU ne troši vreme na čekanje svakog bajta.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).
  - Kontroler prevodi komandu u električne signale i protokol uređaja.
  - Kada je gotovo, javlja CPU-u preko prekida.
- Prednost: CPU ne troši vreme na čekanje svakog bajta.

Uloga kontrolera i prekida

- CPU *ne* razgovara direktno sa diskom ili mrežnom karticom.
- Umesto toga:
  - CPU šalje komandu **kontroleru** (npr. „pročitaj sektor 12345“).
  - Kontroler prevodi komandu u električne signale i protokol uređaja.
  - Kada je gotovo, javlja CPU-u preko prekida.
- Prednost: CPU ne troši vreme na čekanje svakog bajta.

Disk kontroler optimizuje redosled čitanja sektora, koristi sopstveni keš i minimizuje pomeranje glave $\rightarrow$ daleko bolje iskorišćenje diska.

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.
- **NVMe** – protokol za SSD preko PCIe:

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.
- **NVMe** – protokol za SSD preko PCIe:
  - hiljade paralelnih redova komandi,

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.
- **NVMe** – protokol za SSD preko PCIe:
  - hiljade paralelnih redova komandi,
  - vrlo mala latencija (mikrosekunde),

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.
- **NVMe** – protokol za SSD preko PCIe:
  - hiljade paralelnih redova komandi,
  - vrlo mala latencija (mikrosekunde),
  - propusnost više GB/s po uređaju.

Standardi i interfejsi: USB, PCIe, NVMe

- **USB** – univerzalna serijska magistrala:
  - tastature, miševi, kamere, fleš diskovi...
  - novije verzije: USB 3.2, USB4 – desetine Gb/s.
  - prenosi i **podatke** i **napajanje**.
- **PCIe** – glavni „autoput“ za GPU, NVMe, mrežu.
- **NVMe** – protokol za SSD preko PCIe:
  - hiljade paralelnih redova komandi,
  - vrlo mala latencija (mikrosekunde),
  - propusnost više GB/s po uređaju.
- Kombinacija PCIe + NVMe je standard za brze diskove danas.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.
- **Drajveri** – znaju detalje konkretnog uređaja i kontrolera.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.
- **Drajveri** – znaju detalje konkretnog uređaja i kontrolera.
- **Kontroleri** – prevode komande u signale, rade DMA prenose.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.
- **Drajveri** – znaju detalje konkretnog uređaja i kontrolera.
- **Kontroleri** – prevode komande u signale, rade DMA prenose.
- **Fizički uređaji** – disk ploče, flash čipovi, mrežni PHY, itd.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.
- **Drajveri** – znaju detalje konkretnog uređaja i kontrolera.
- **Kontroleri** – prevode komande u signale, rade DMA prenose.
- **Fizički uređaji** – disk ploče, flash čipovi, mrežni PHY, itd.

Slojna organizacija I/O sistema

- **Aplikacije** – čitaju i pišu fajlove, komuniciraju preko mreže.
- **Operativni sistem** – fajl sistem, mrežni stek, I/O podsystem.
- **Drajveri** – znaju detalje konkretnog uređaja i kontrolera.
- **Kontroleri** – prevode komande u signale, rade DMA prenose.
- **Fizički uređaji** – disk ploče, flash čipovi, mrežni PHY, itd.

Ovakva hijerarhija omogućava da aplikacije rade isto, bez obzira na konkretan model SSD-a, mrežne kartice ili tastature.

Integracija na matičnoj ploči

- **Matična ploča** = fizička osnova sistema:

Savremeni CPU već sadrži memorijski kontroler i deo PCIe linija – ostatak funkcionalnosti je u PCH čipu.

Integracija na matičnoj ploči

- **Matična ploča** = fizička osnova sistema:
  - nosi CPU, RAM module, slotove za GPU/NVMe,

Savremeni CPU već sadrži memorijski kontroler i deo PCIe linija – ostatak funkcionalnosti je u PCH čipu.

Integracija na matičnoj ploči

- **Matična ploča** = fizička osnova sistema:
  - nosi CPU, RAM module, slotove za GPU/NVMe,
  - vodi komunikacione linije (PCIe, memorijski kanali),

Savremeni CPU već sadrži memorijski kontroler i deo PCIe linija – ostatak funkcionalnosti je u PCH čipu.

Integracija na matičnoj ploči

- **Matična ploča** = fizička osnova sistema:
  - nosi CPU, RAM module, slotove za GPU/NVMe,
  - vodi komunikacione linije (PCIe, memorijski kanali),
  - obezbeđuje napajanje i sinhronizaciju.

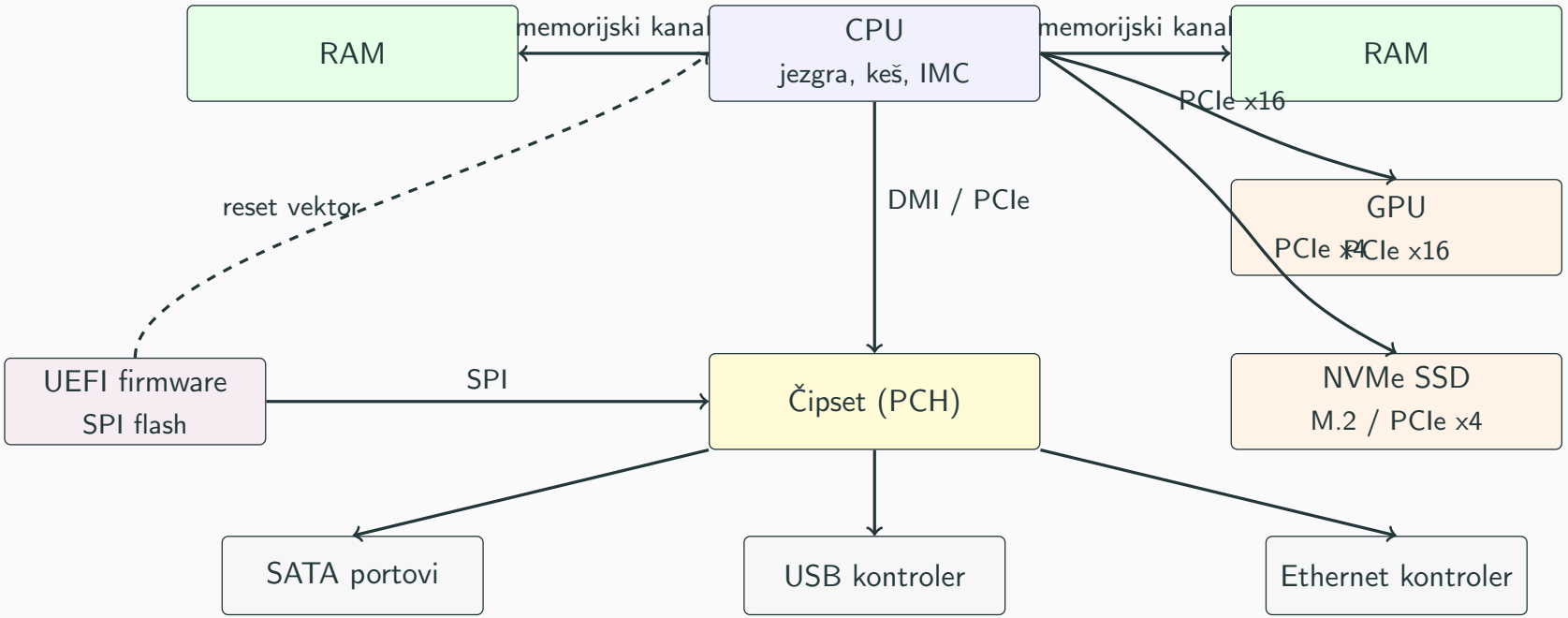
Savremeni CPU već sadrži memorijski kontroler i deo PCIe linija – ostatak funkcionalnosti je u PCH čipu.

Integracija na matičnoj ploči

- **Matična ploča** = fizička osnova sistema:
  - nosi CPU, RAM module, slotove za GPU/NVMe,
  - vodi komunikacione linije (PCIe, memorijski kanali),
  - obezbeđuje napajanje i sinhronizaciju.
- **Čipset (PCH)** upravlja perifernim uređajima, dodatnim PCIe linijama, USB, SATA, mrežom.

Savremeni CPU već sadrži memorijski kontroler i deo PCIe linija – ostatak funkcionalnosti je u PCH čipu.

Integracija na matičnoj ploči



BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.
  - Taj kod je **BIOS** ili savremeniji **UEFI firmware**.

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.
  - Taj kod je **BIOS** ili savremeniji **UEFI firmware**.
- BIOS (klasično):

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.
  - Taj kod je **BIOS** ili savremeniji **UEFI firmware**.
- BIOS (klasično):
  - monolitan, u ROM-u, teško ažuriranje,

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.
  - Taj kod je **BIOS** ili savremeniji **UEFI firmware**.
- BIOS (klasično):
  - monolitan, u ROM-u, teško ažuriranje,
  - radi POST test, inicijalizuje osnovni hardver, pokreće OS.

BIOS i UEFI – ko zapravo „prvi“ radi?

- Kada pritisnemo dugme za uključivanje:
  - CPU počinje da izvršava kod iz male flash memorije na ploči.
  - Taj kod je **BIOS** ili savremeniji **UEFI firmware**.
- BIOS (klasično):
  - monolitan, u ROM-u, teško ažuriranje,
  - radi POST test, inicijalizuje osnovni hardver, pokreće OS.
- Nedostaci: ograničena veličina, slaba podrška za nove uređaje, velike diskove, mrežu...

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,
  - da pokreće pomoćne aplikacije pre OS-a.

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,
  - da pokreće pomoćne aplikacije pre OS-a.
- Podržava *Secure Boot*, veće diskove, moderni hardver.

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,
  - da pokreće pomoćne aplikacije pre OS-a.
- Podržava *Secure Boot*, veće diskove, moderni hardver.
- Može se ažurirati (flešovanje BIOS/UEFI-ja).

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,
  - da pokreće pomoćne aplikacije pre OS-a.
- Podržava *Secure Boot*, veće diskove, moderni hardver.
- Može se ažurirati (flešovanje BIOS/UEFI-ja).

UEFI – savremeni firmware

- **UEFI** = modularan firmware u malom flash čipu (SPI flash).
- Može:
  - da koristi mrežne drajvere,
  - da prikaže grafički interfejs,
  - da pokreće pomoćne aplikacije pre OS-a.
- Podržava *Secure Boot*, veće diskove, moderni hardver.
- Može se ažurirati (flešovanje BIOS/UEFI-ja).

UEFI inicijalizuje hardver, pronalazi uređaj za boot, učitava **boot loader**, i tek tada kontrolu preuzima operativni sistem.

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.
- Pojmovi:

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.
- Pojmovi:
  - vršna snaga (peak power)

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.
- Pojmovi:
  - vršna snaga (peak power)
  - TDP – *Thermal Design Power*

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.
- Pojmovi:
  - vršna snaga (peak power)
  - TDP – *Thermal Design Power*
  - energija po zadatku (energy per task)

Trendovi: energija kao ograničenje

- Više nije problem „da li možemo da dodamo tranzistore“, već: koliko njih sme da bude upaljeno odjednom.
- Pojmovi:
  - vršna snaga (peak power)
  - TDP – *Thermal Design Power*
  - energija po zadatku (energy per task)
- Često je bolje: uraditi posao brzo, pa spustiti sve u sleep, nego stalno raditi „polako“.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.
- **DVFS** – podešavanje napona i frekvencije u hodu.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.
- **DVFS** – podešavanje napona i frekvencije u hodu.
- **Power gating** – potpuno isključivanje napajanja delovima čipa.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.
- **DVFS** – podešavanje napona i frekvencije u hodu.
- **Power gating** – potpuno isključivanje napajanja delovima čipa.
- **Race-to-halt** – radi maksimalno brzo pa spava umesto da radi stalno polovičnom brzinom.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.
- **DVFS** – podešavanje napona i frekvencije u hodu.
- **Power gating** – potpuno isključivanje napajanja delovima čipa.
- **Race-to-halt** – radi maksimalno brzo pa spava umesto da radi stalno polovičnom brzinom.

Tehnike upravljanja energijom

- **Clock gating** – isključivanje takta neaktivnim blokovima.
- **DVFS** – podešavanje napona i frekvencije u hodu.
- **Power gating** – potpuno isključivanje napajanja delovima čipa.
- **Race-to-halt** – radi maksimalno brzo pa spava umesto da radi stalno polovičnom brzinom.

Ne mogu svi tranzistori istovremeno da rade na maksimalnoj frekvenciji zbog termičkih ograničenja $\rightarrow$ deo čipa je često „u mraku“ (neaktivan).

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).
- **Akceleratori** (TPU, NPU, Tensor Cores...):

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).
- **Akceleratori** (TPU, NPU, Tensor Cores...):
  - specijalizovani za zadatke (matrične operacije, ML, video kodiranje),

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).
- **Akceleratori** (TPU, NPU, Tensor Cores...):
  - specijalizovani za zadatke (matrične operacije, ML, video kodiranje),
  - postižu ogromne performanse po vatu.

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).
- **Akceleratori** (TPU, NPU, Tensor Cores...):
  - specijalizovani za zadatke (matrične operacije, ML, video kodiranje),
  - postižu ogromne performanse po vatu.

SoC i domen-specifični akceleratori

- **SoC** (System-on-Chip):
  - CPU jezgra, GPU, memorijski kontroler, mreža, I/O – sve na jednom čipu.
  - Kratke veze, manja potrošnja, bolja integracija (telefoni, tableti, laptopovi).
- **Akceleratori** (TPU, NPU, Tensor Cores...):
  - specijalizovani za zadatke (matrične operacije, ML, video kodiranje),
  - postižu ogromne performanse po vatu.

Dalji razvoj ne ide samo kroz „još GHz“, već kroz paralelizam, heterogenost i energetsku efikasnost.