RAČUNALNIŠKA ARHITEKTURA

3 Osnovni principi delovanja

3 Osnovni principi delovanja - vsebina

- Amdahlov zakon
- Jeziki, nivoji in navidezni računalniki
  - Računalnik kot zaporedje navideznih računalnikov
  - Prehajanje iz jezika J2 v jezik J1
  - Strojna in programska oprema računalnika

Računanje in izračunljivost

- Definicija računanja: določanje vrednosti neki funkciji
- $y = f(x)$
  - x – vhodni podatek (število, datoteka, zvočni signal, slika ...)
  - y – rezultat, ki ga želimo izračunati
  - f – izračun, pisanje na datoteko, obdelava zvoka, slike ...

- Neka funkcija *f(x)* je izračunljiva, če obstaja postopek, s katerim lahko določimo njeno vrednost *y* za vse podatke *x*, za katere je definirana.

- Postopku ali navodilu, kako izračunati vrednost funkcije, pravimo algoritem.

- Neki problem je izračunljiv, če obstaja algoritem za izračun; če algoritem ne obstaja, je problem neizračunljiv.

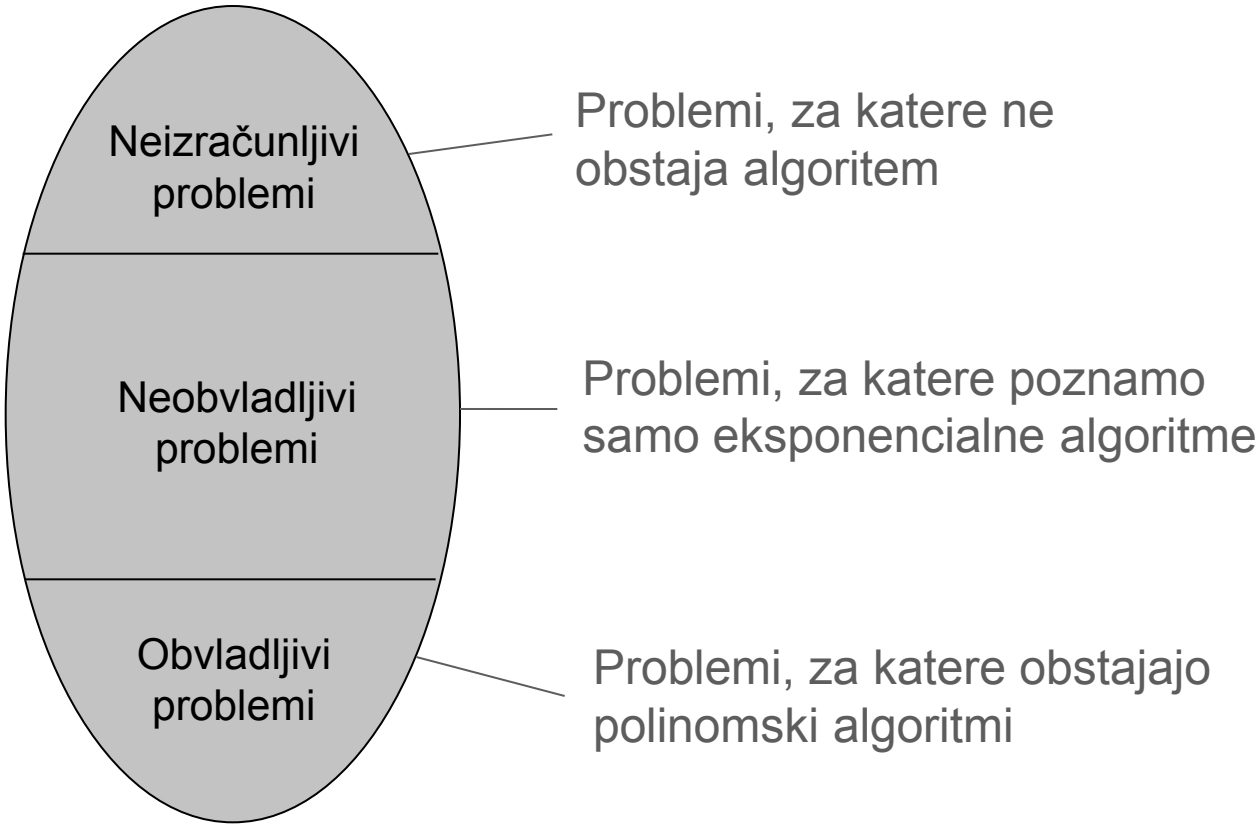
- Teoretični modeli računanja – vsak problem, za katerega v takem modelu obstaja algoritem, je izračunljiv.
  - Lambda račun (A. Curch, S. C. Kleene, J. B. Rosser)
  - Turingov stroj (Alan Turing)
  - Produkcijski sistem (E. Post)
  - Vsi objavljeni leta 1936
  - . . .
- Neki problem je izračunljiv, če ga je v končnem številu korakov mogoče izračunati na nekem Turingovem stroju (Church-Turingova hipoteza).

- **Turingov stroj:** matematični (miselni) model računanja - to ni resničen stroj.

- **Von Neumannov model** ima dvojni pomen:
  - matematični (teoretični) model računanja
  - resničen stroj

- Von Neumannov računalniški model je ekvivalenten (za praktične namene) Turingovemu stroju $\Rightarrow$ na njem se da izračunati vse, kar je izračunljivo.

Razdelitev množice vseh problemov na tri glavne podmnožice

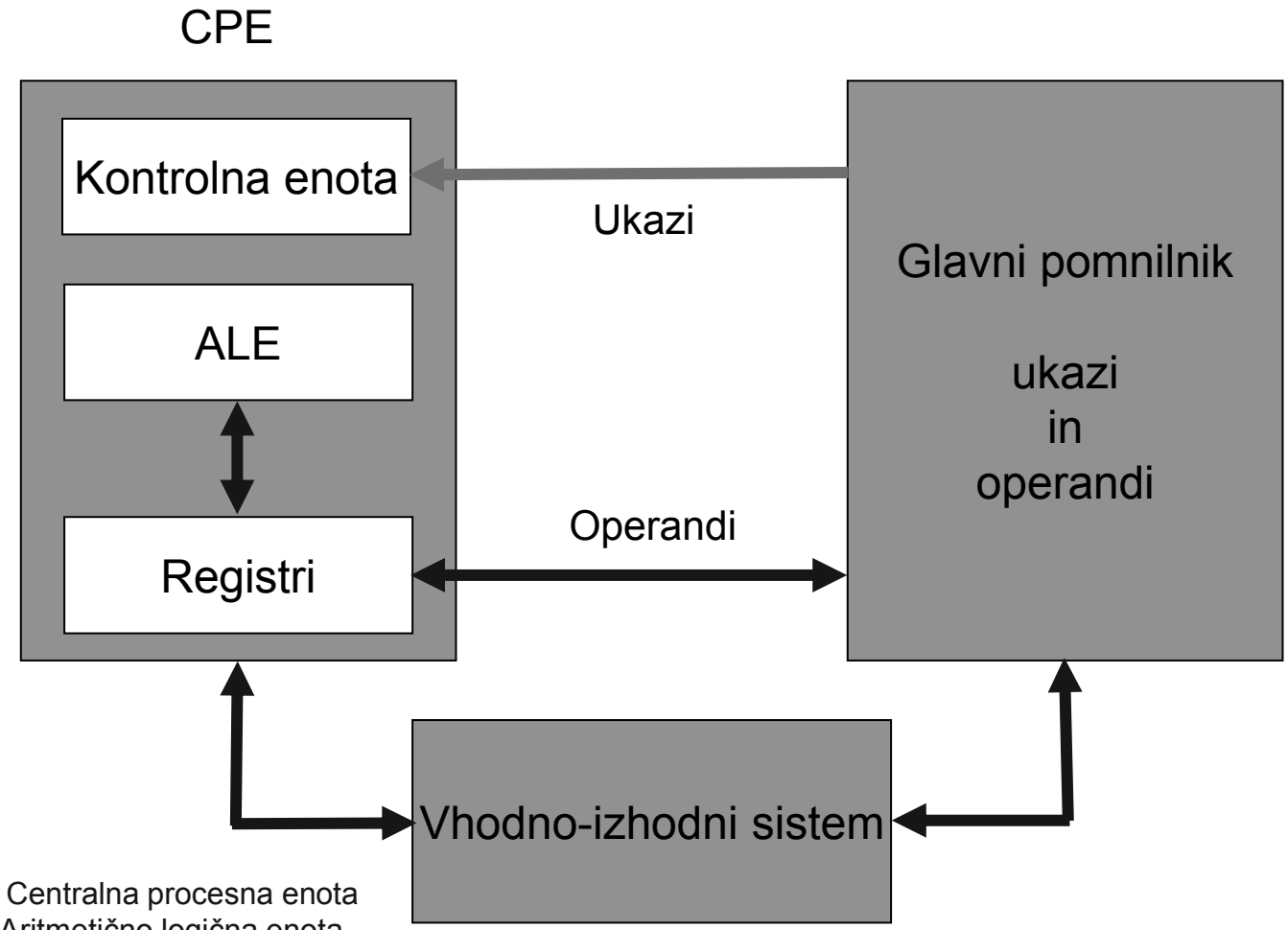


- von Neumannov model računanja
- von Neumannov računalniški model
- von Neumannov računalnik
- von Neumannova arhitektura

3.1 Von Neumannov računalniški model

- Sestavljajo ga trije osnovni deli:
  - CPE (centralna procesna enota)
  - Glavni pomnilnik
  - V/I sistem
- Je stroj s shranjenim programom, ki je shranjen v glavnem pomnilniku. Ukazi v programu določajo, kaj bo stroj delal.
- **Program vodi delovanje stroja** .
- CPE jemlje ukaze iz glavnega pomnilnika in jih izvaja drugega za drugim.

Von Neumannov računalniški model



CPE – Centralna procesna enota
ALE – Aritmetično logična enota

- **CPE** iz glavnega pomnilnika jemlje ukaze in jih izvršuje. V današnjih računalnikih je poleg CPE še več procesorjev, zato oznaka **centralna.** Delimo jo na tri dele:

  - KONTROLNA ENOTA - skrbi za prevzemanje ukazov in operandov in aktiviranje operacij, ki so določene z ukazi.
  - ALE - izvaja aritmetične operacije (seštevanje . . .) in logične operacije (AND . . .).
  - REGISTRI – več povezanih pomnilniških celic, ki služijo za shranjevanje vrednosti.
    - Programsko nedostopni registri – potrebni za delovanje CPE.
    - Programsko dostopni registri (arhitekturni registri) za shranjevanje operandov. Predstavljajo majhen in hiter pomnilnik v CPE.

- **Glavni pomnilnik** je sestavljen iz pomnilniških besed. Vsaka pomnilniška beseda ima svoj enolični naslov.
  - V njem so shranjeni ukazi in operandi.
  - Oznaka glavni zopet služi za razlikovanje od drugih pomnilnikov v današnjih računalnikih (predpomnilniki, navidezni pomnilnik).
- **V/I sistem** za prenos informacije v zunanji svet ali iz zunanjega sveta. Informacija je v CPE in glavnem pomnilniku shranjena v obliki, ki ni dostopna zunanjemu svetu.
  - Sestavni del V/I sistema so vhodno-izhodne naprave, ki pretvarjajo informacijo v neko drugo obliko, ki je primerna za uporabnika ali pa služijo kot pomožni pomnilniki.

Delovanje von Neumannovega računalnika

- Njegovo delovanje popolnoma določajo ukazi (strojni ukazi), ki jih CPE jemlje iz glavnega pomnilnika zaporedoma enega za drugim.
- Strojni ukazi so v pomnilniku shranjeni eden za drugim po naraščajočih naslovih.
- Na neki način je določeno, iz katerega naslova se vzame prvi ukaz po vklopu računalnika ali po pritisku na tipko RESET.
  - Najenostavneje: prva ali zadnja pomnilniška lokacija – najnižji ali najvišji naslov v pomnilniku.

Pri vsakem ukazu razlikujemo dva koraka

- **1. korak:** Jemanje ukaza iz pomnilnika
  (tudi branje ali prevzem ukaza)
  - ukazno prevzemni cikel
  - angl. fetch cycle

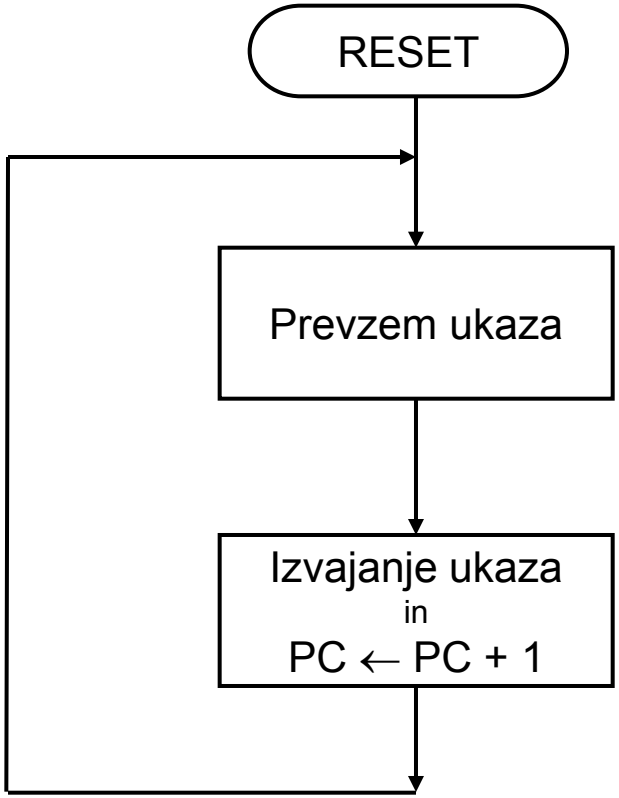
- V CPE je poseben register - programski števec (PC - Program Counter), ki vedno vsebuje pomnilniški naslov, na katerem je v pomnilniku shranjen naslednji ukaz.

- **2. korak:** Izvrševanje v 1. koraku prevzetega ukaza
  - izvršilni cikel
  - angl. execute cycle

- Vsak ukaz vsebuje dve vrsti informacij:
  - informacijo o operaciji, ki naj se izvrši,
  - informacije o operandih, nad katerimi naj se operacija izvrši.

- CPE izvrši operacijo in poskrbi, da je v PC naslov naslednjega ukaza tako, da poveča vsebino PC-ja za 1.

- **Pravilo:** ukazi v pomnilniku so shranjeni po naraščajočih naslovih zato PC $\leftarrow$ PC + 1. To pravilo je rezultat dogovora in določa vrstni red izvajanja ukazov.



1. korak

2. korak

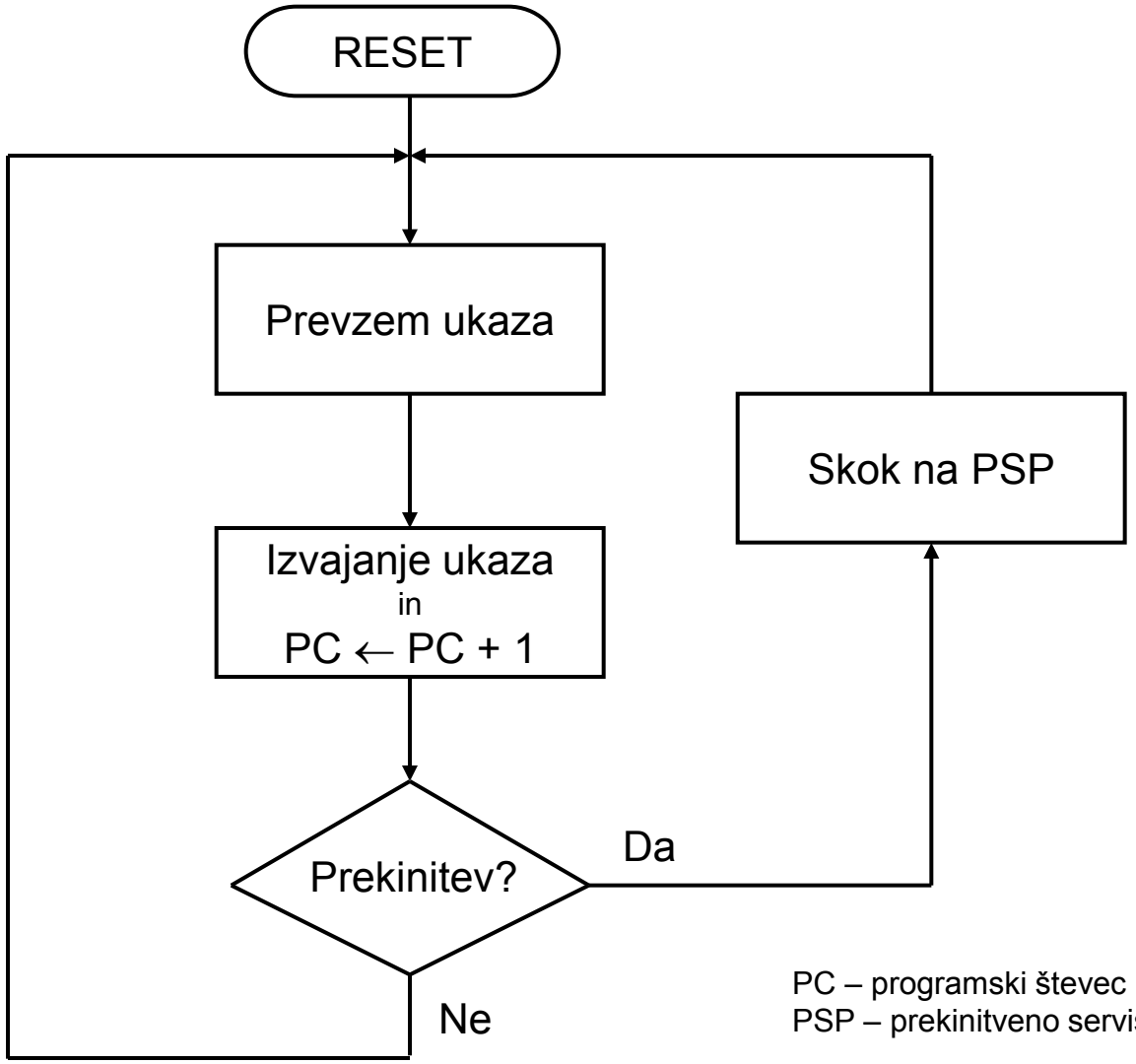
- **Izjema 1:** Skočni ukazi, s katerimi lahko v PC zapišemo poljuben naslov.

- Po zaključku koraka 2 začne CPE zopet s korakom 1.

- Ta dva koraka se ponavljata, dokler računalnik deluje.

- **Izjema 2:** Prekinitev ali past

  CPE po koraku 2 ne prevzame ukaza po pravilu PC $\leftarrow$ PC + 1, temveč začne izvajati drug program - prekinitveno servisni program (PSP).



PC – programski števec
PSP – prekinitveno servisni program

- Zaporedno izvajanje ukazov je počasno in predstavlja osnovno slabost von Neumannovih računalnikov.

- Razširitve osnovnega von Neumannovega modela so zajete v Flynnovi klasifikaciji iz leta 1966.

3.2 Flynnova klasifikacija

- Osnovna kriterija Flynnove klasifikacije za razvrstitev računalnikov:
  - število ukazov, ki se izvršujejo naenkrat (instruction stream),
  - število operandov, ki jih en ukaz naenkrat obdeluje (data stream).
- Po teh kriterijih vsak računalnik spada v enega od štirih razredov:
- 1 SISD (Single Instruction Single Data)
  - klasični Von Neumannovi računalniki
  - Intel Pentium 4

- 2 SIMD (Single Instruction Multiple Data)
  - vektorski računalniki (paralelni računalniki)
  - ukazi SSE (Streaming SIMD Extensions) pri procesorjih x86
- 3 MISD ! (Multiple Instruction Single Data)
  - Danes ne obstaja noben tak računalnik.
- 4 MIMD (Multiple Instruction Multiple Data)
  - multiprocesorski računalniki (paralelni računalniki)
  - npr. Intel Xeon e5345 (Clovertown), AMD Opteron X4 (Barcelona) . . .

- Pri MIMD računalnikih se naenkrat izvaja več ukazov, vsak na svojih operandih.
- MIMD računalnik tvori več povezanih navadnih von Neumannovih računalnikov – več CPE, ki so med seboj povezane.
- Večjedrne računalnike včasih štejemo tudi kar med SISD, čeprav lahko današnje večjderne mikroprocesorje uvrščamo tudi v SIMD in MIMD.

3.3 Glavni pomnilnik v von Neumannovem računalniku

- **Definicija**
  - Glavni pomnilnik je **pasivna naprava** in služi kot skladišče za shranjevanje ukazov in operandov.
  - Osnovna celica v pomnilniku je enobitna pomnilniška celica, ki lahko hrani 1 bit informacije (vsebina 0 ali 1).

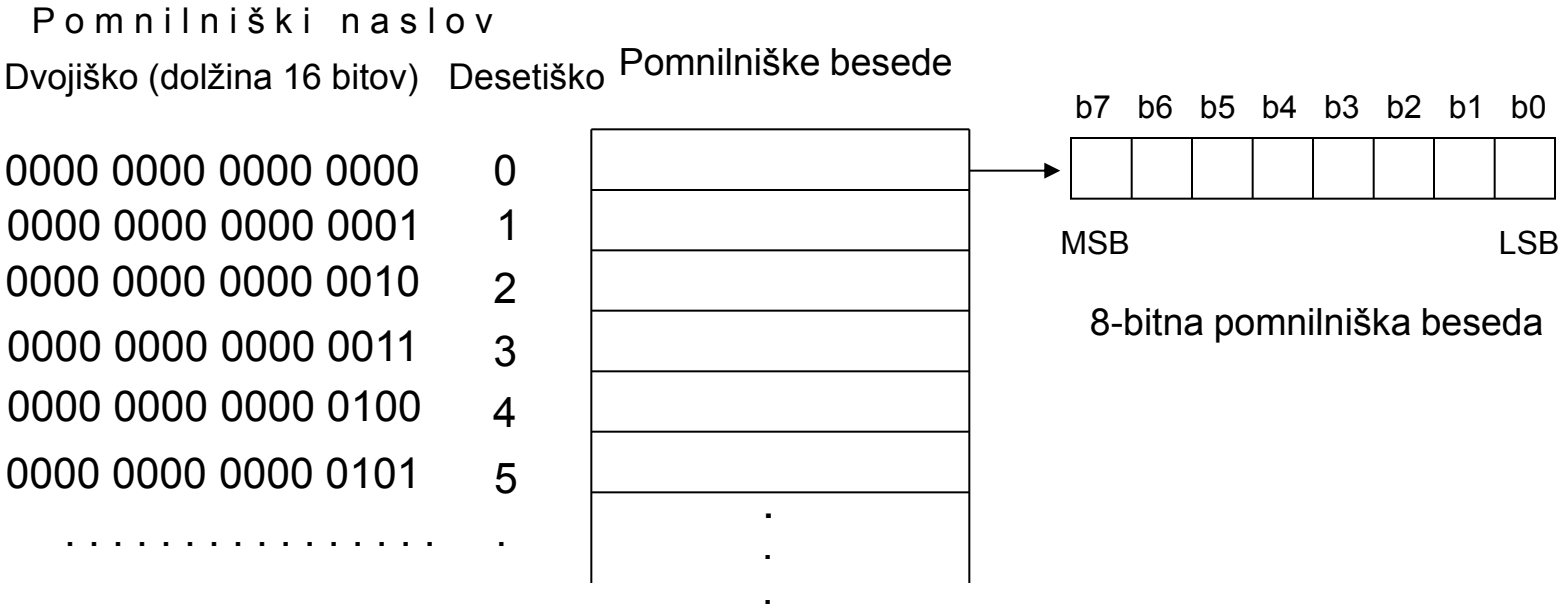
- **Pomnilniška beseda** (tudi pomnilniška lokacija)
  - Pomnilnik je enodimenzionalno zaporedje pomnilniških besed.
  - Pomnilniško besedo sestavlja določeno število enobitnih pomnilniških celic.
  - **Dolžina pomnilniške besede** je število enobitnih pomnilniških celic, ki sestavljajo pomnilniško besedo. Danes je najpogostejša dolžina besede 1 bajt (= 8 bitov).

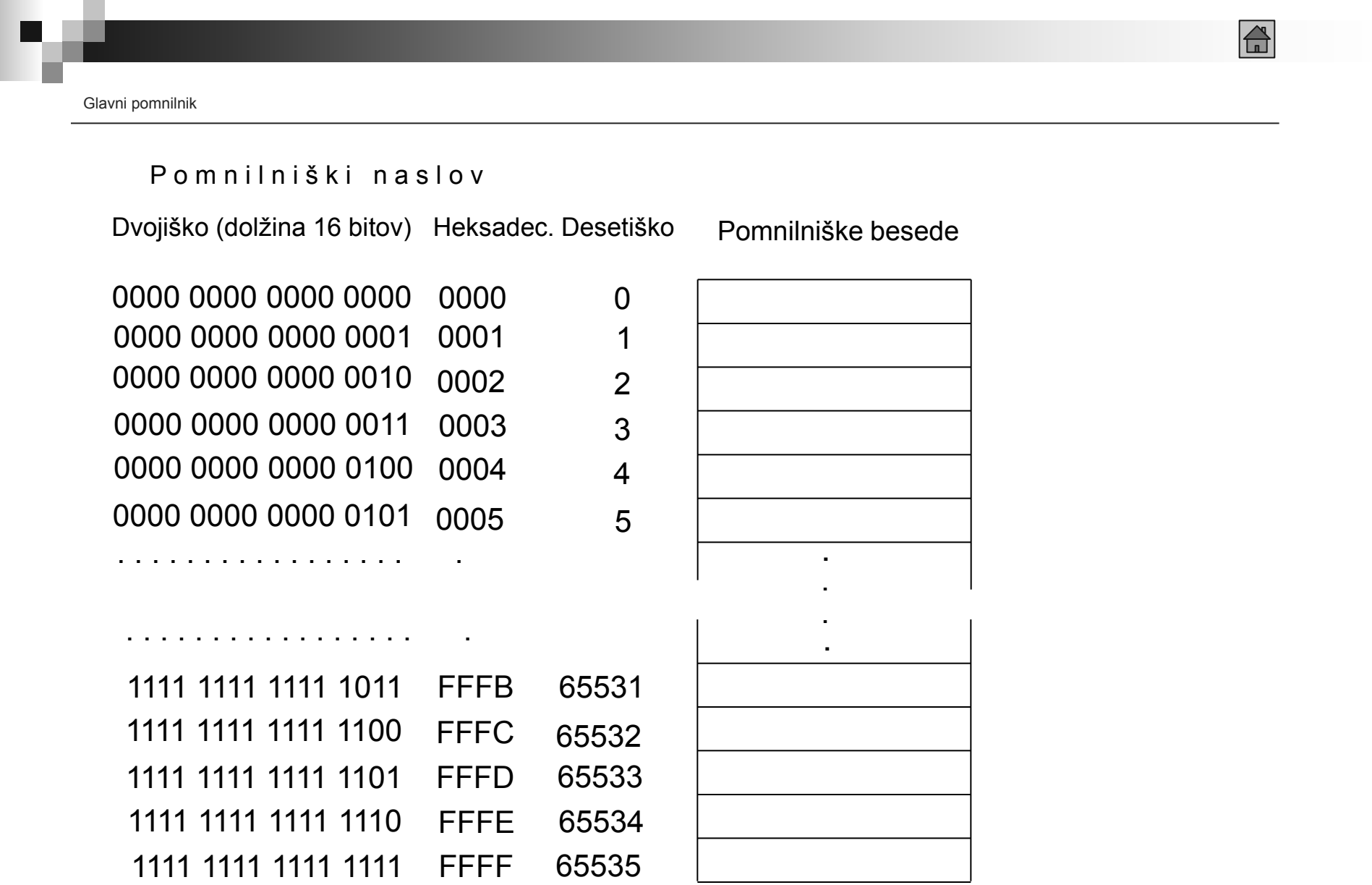
- Pomnilniška beseda je definirana kot najmanjše število bitov, ki imajo svoj naslov. Pomnilniška beseda je torej najmanjša naslovljiva enota v pomnilniku.

- **Pomnilniški naslov**
  - Vsaka pomnilniška beseda ima svoj enolični pomnilniški naslov.
  - Število bitov, ki sestavljajo naslov, imenujemo **dolžina naslova**.
  - Dolžina naslova v bitih določa naslovni prostor.

- **Naslovni prostor** (tudi pomnilniški prostor) je množica vseh naslovov in določa tudi največjo možno velikost pomnilnika.

- **Vsebina pomnilniške besede** se lahko spreminja. V 8-bitno pomnilniško besedo lahko shranimo $2^8 = 256$ različnih vsebin.
- Naslov pomnilniške besede je nespremenljiv.
- Število pomnilniških besed v glavnem pomnilniku ni nujno enako velikosti naslovnega prostora.
- Deli naslovnega prostora so lahko prazni (vsi naslovi niso uporabljeni) $\Rightarrow$ glavni pomnilnik je manjši od največje možne velikosti.



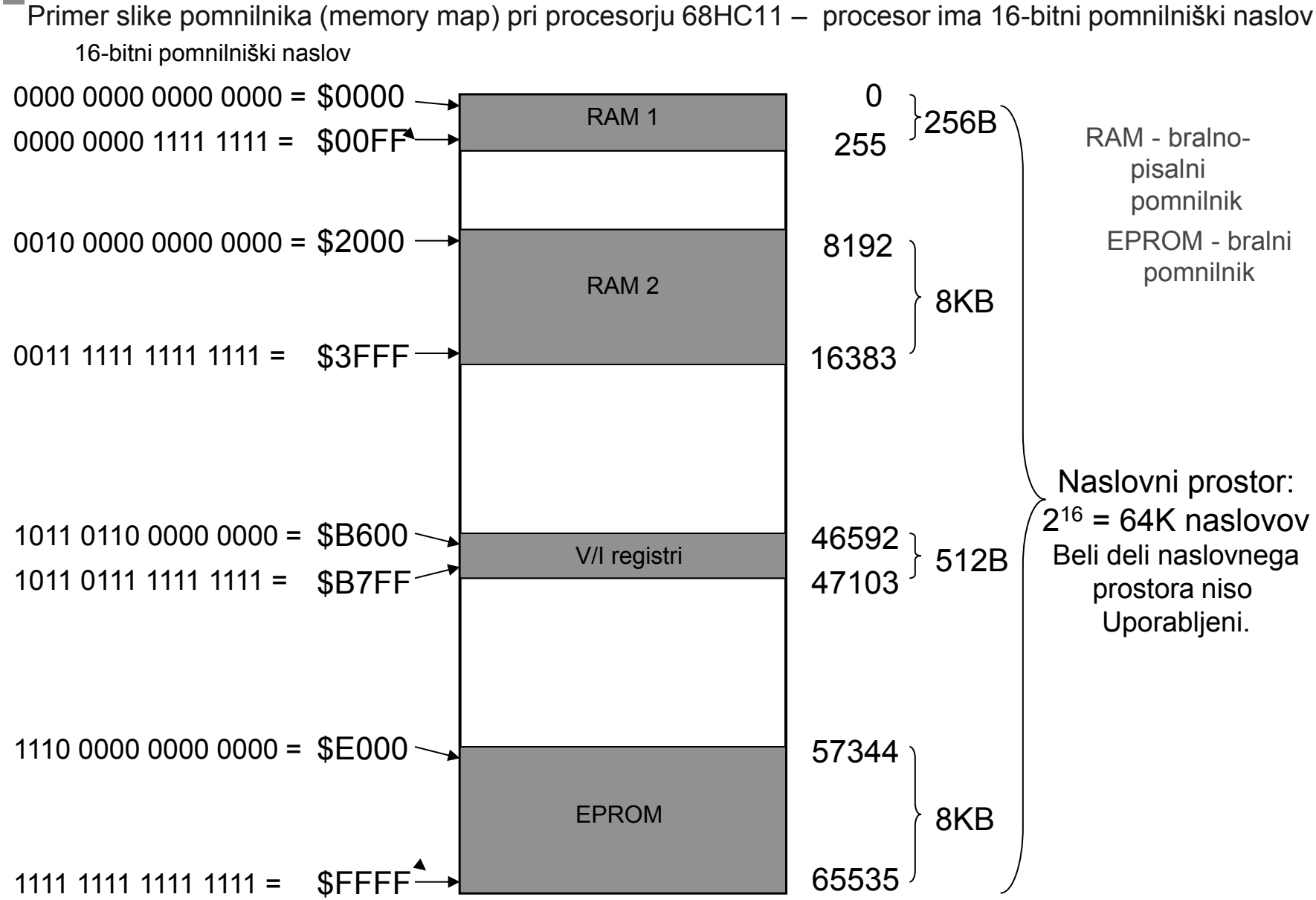


Predpone kilo, mega, giga idr. so samo pri velikosti pomnilnika potence števila 2!

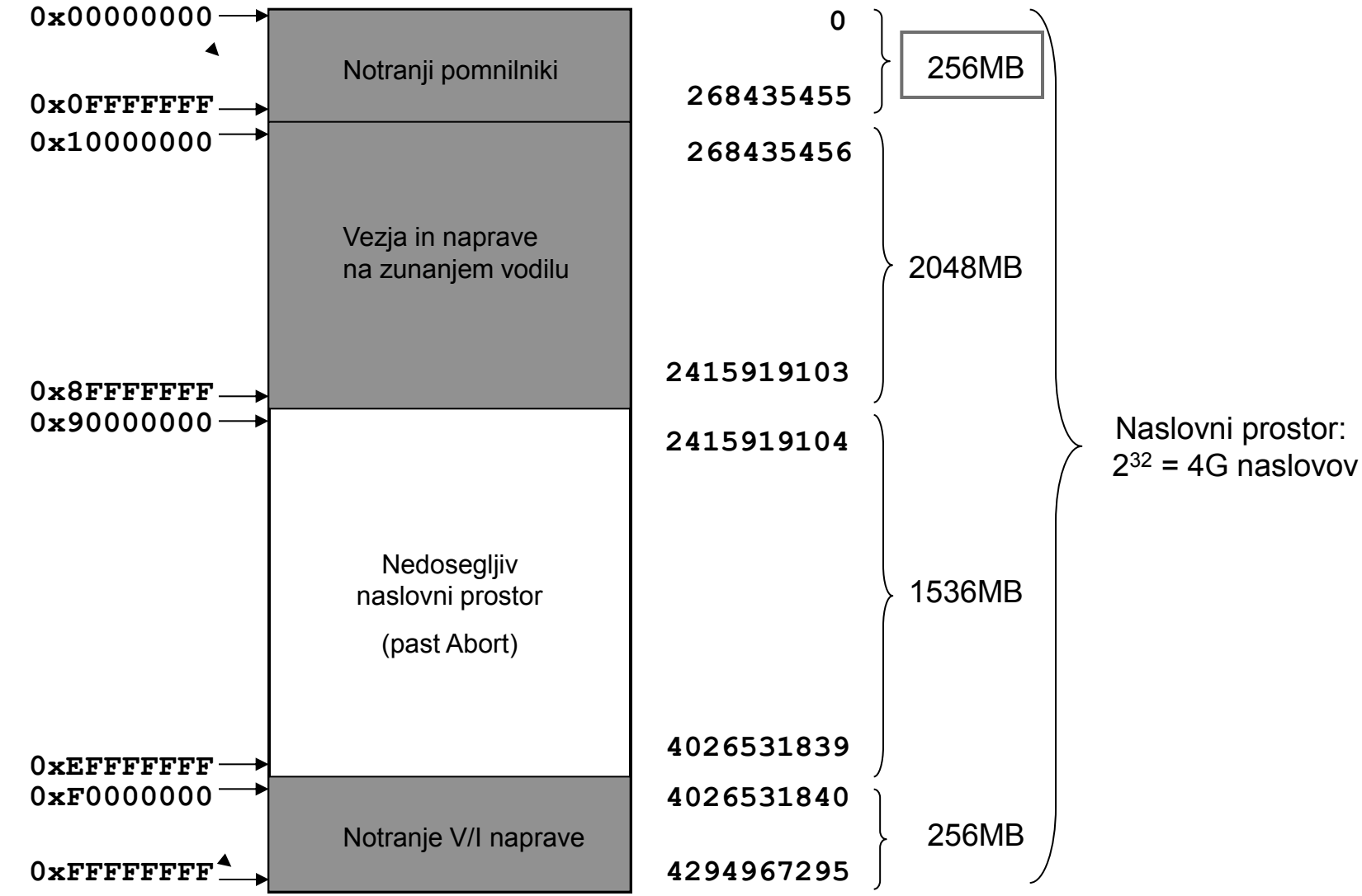
- 1K (kilo) = 2^{10} = 1024 (1 KB = 1024 B)
- 1M (mega) = 2^{20} = 1 048 576 (1 MB = 1 048 576 B)
- 1G (giga) = 2^{30} = 1 073 741 824
  - Vzrok je tehnološki: 20-bitni pomnilniški naslov omogoča 2^{20} različnih naslovov.
  - KiB = 2^{10} B, MiB = 2^{20} B, GiB = 2^{30} B predlog IEC 1998

- Druga področja (frekvenca, hitrost prenosa …)
- 1k = 10^3 = 1000 (1 km = 1000 m)
- 1M = 10^6 = 1,000.000 (1 MB/s=1,000.000 B/s)
- 1G = 10^9 = 1,000.000.000

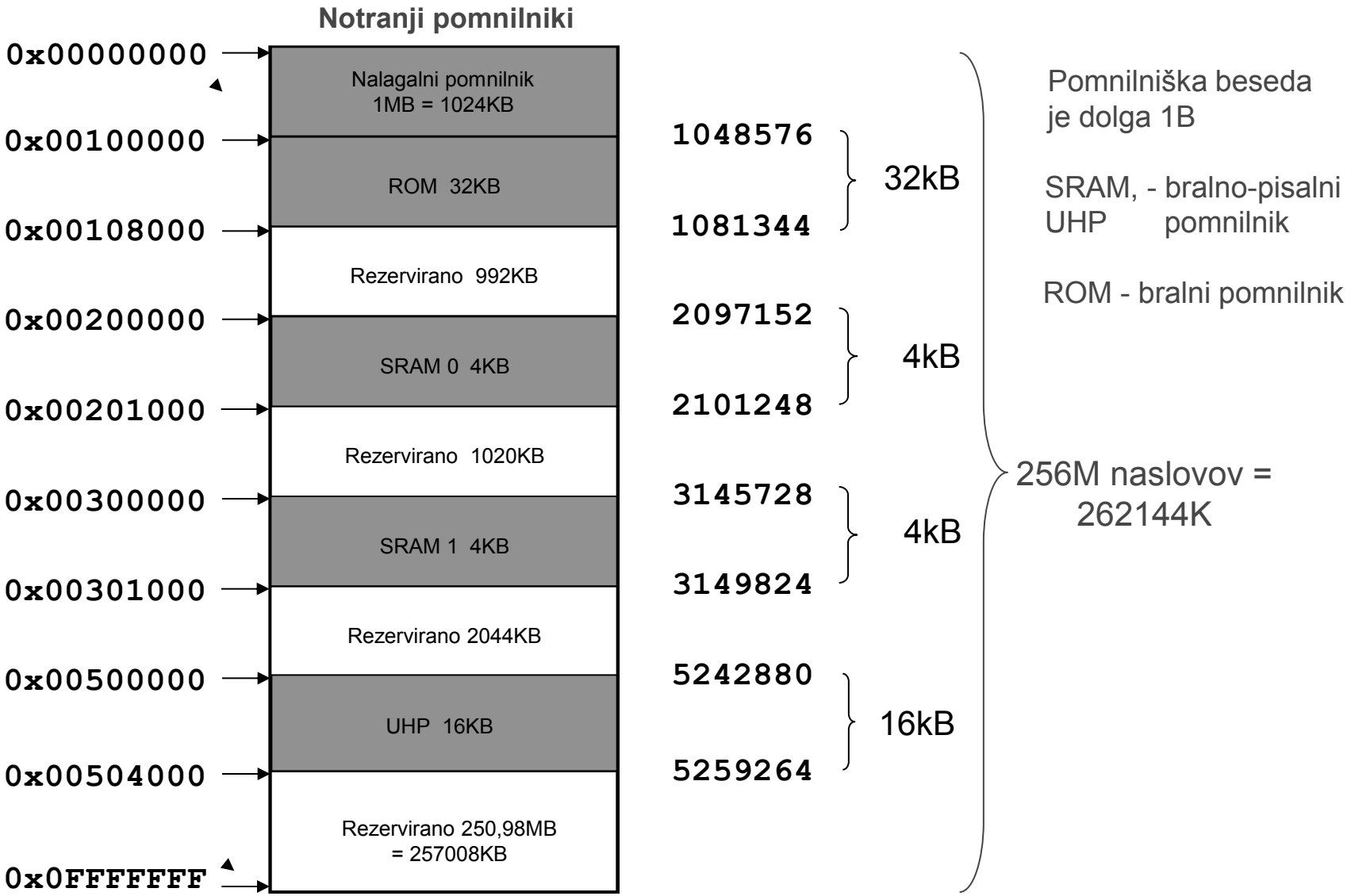
Primer slike pomnilnika (memory map) pri procesorju 68HC11 – procesor ima 16-bitni pomnilniški naslov



Primer slike pomnilnika pri procesorju AT91SAM9260 (32-bitni pomnilniški naslov)



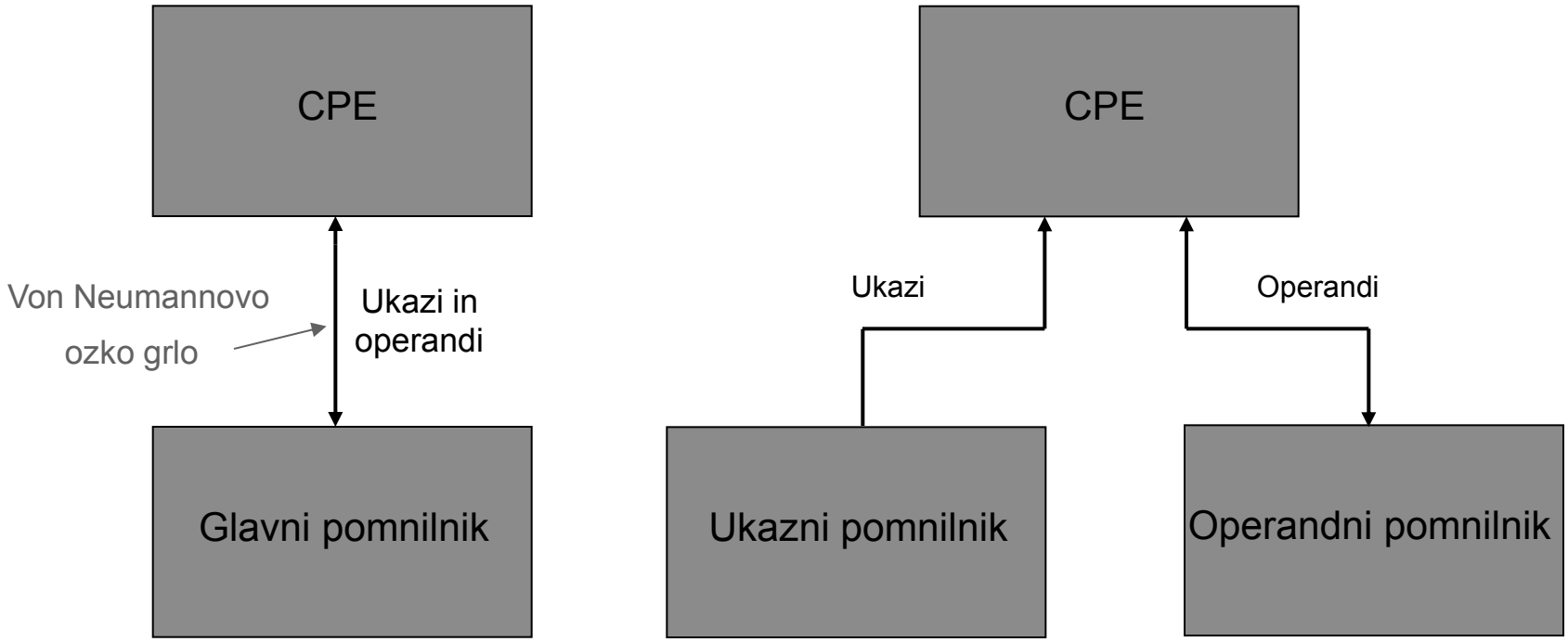
Slika notranjega pomnilnika (prvih 256 MB) do pri AT91SAM9260



Von Neumannovo ozko grlo

- Prenosi CPE $\leftrightarrow$ gl. pomnilnik - promet
- Von Neumannovo ozko grlo - povezava med CPE in glavnim pomnilnikom
- Eden od načinov za razširitev tega ozkega grla je razdelitev glavnega pomnilnika v dva dela.

Razširitev von Neumannovega ozkega grla



Princetonska pomnilniška arhitektura

Harvardska pomnilniška arhitektura

- Pomnilnik je pri harvardski arhitekturi razdeljen na dva ločena pomnilnika.
- V enem so shranjeni samo operandi – operandni pomnilnik, v drugem pa samo ukazi – ukazni pomnilnik.
- Ukazni in operandni pomnilnik lahko delujeta istočasno. Tako lahko dosežemo do dvakrat večjo hitrost.
- Harvardska arhitektura se danes uporablja pri predpomnilniku na najnižjem nivoju (operandni in ukazni predpomnilnik), glavni pomnilnik pa je pri večini računalnikov en sam (princetonska arhitektura).

Dostop do pomnilnika

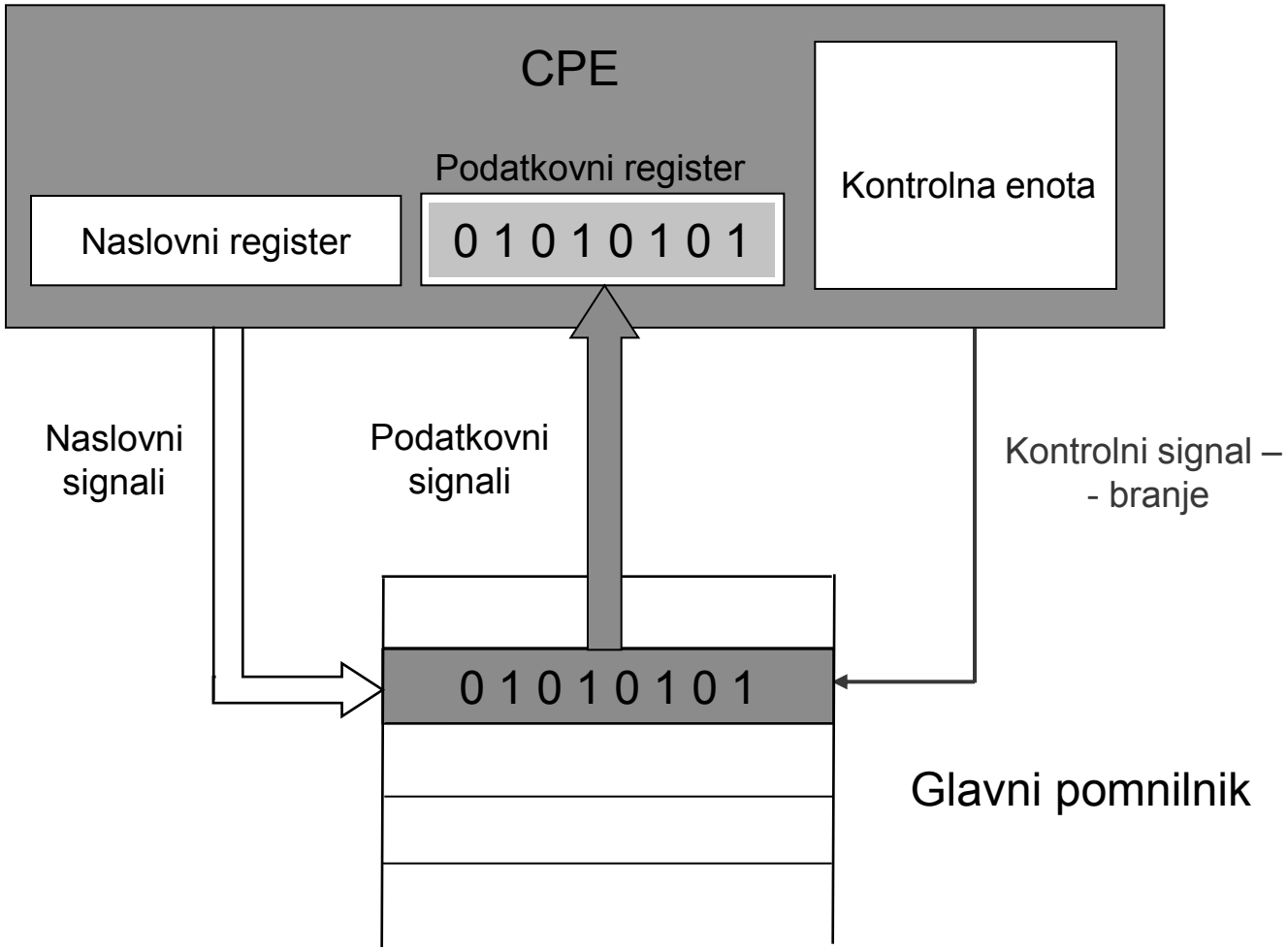
- CPE dostopa do pomnilniške besede tako, da v pomnilnik pošlje naslov te besede in signal za smer prenosa.

- Smer prenosa - vrsta dostopa

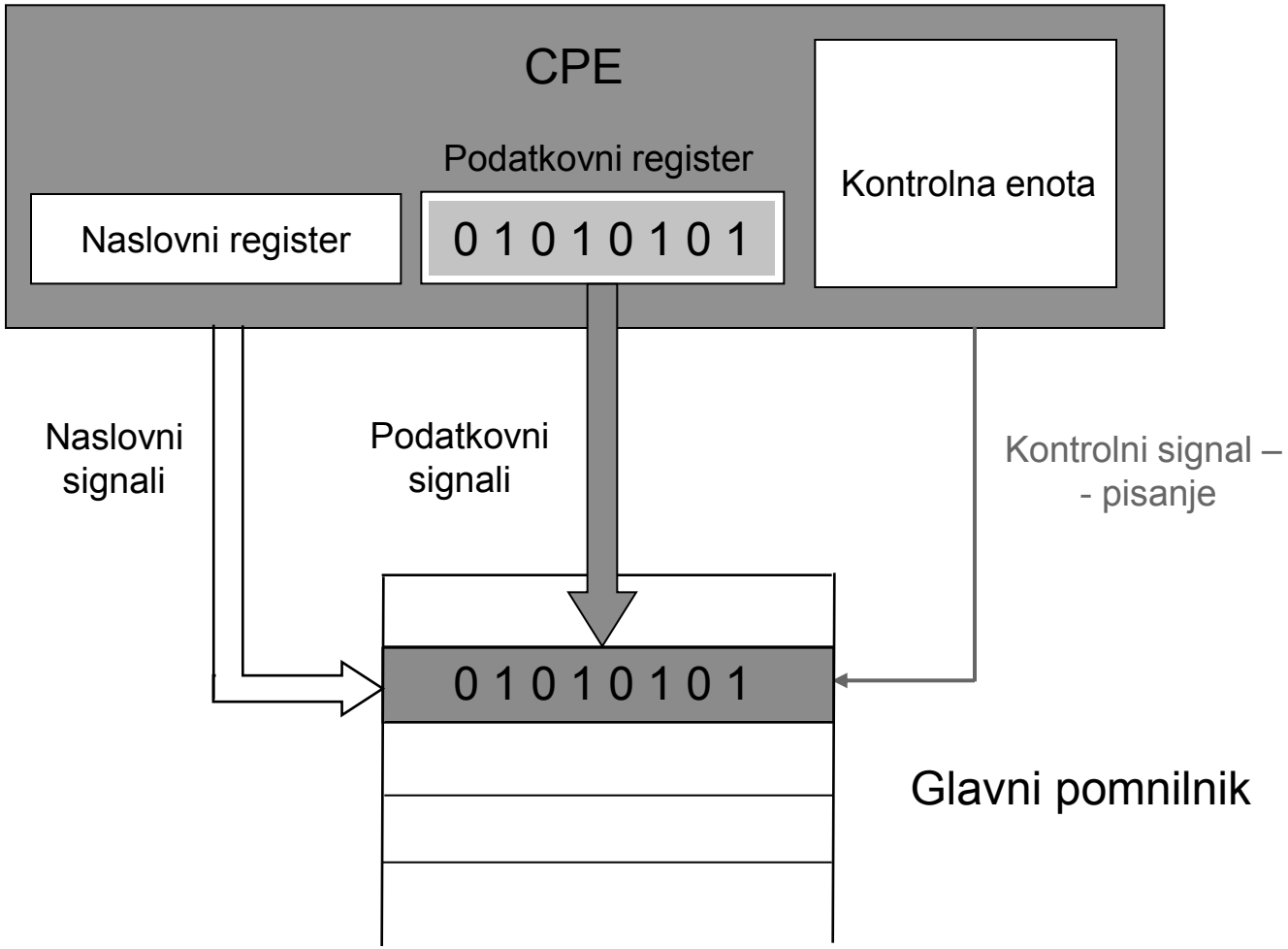
  - CPE $\leftarrow$ gl. pomnilnik - branje (bralni dostop)

  - CPE $\rightarrow$ gl. pomnilnik - pisanje (pisalni dostop)

Povezava med CPE in glavnim pomnilnikom – bralni dostop



Povezava med CPE in glavnim pomnilnikom – pisalni dostop



Povzetek lastnosti glavnega pomnilnika v von Neumannovem računalniku

- Pomnilnik je enodimenzionalen in organiziran v besede. Vsaka beseda ima svoj naslov.
- Ni razlike med ukazi in operandi.
- Pomen ni sestavni del operandov.
- Veliko več bralnih kot pisalnih dostopov
  - Razmerje: okrog 80 % branj, 20 % pisanj
  - Zakaj?

Kombinacija 8 bitov v pomnilniku, npr. 1000 1011, lahko pomeni:

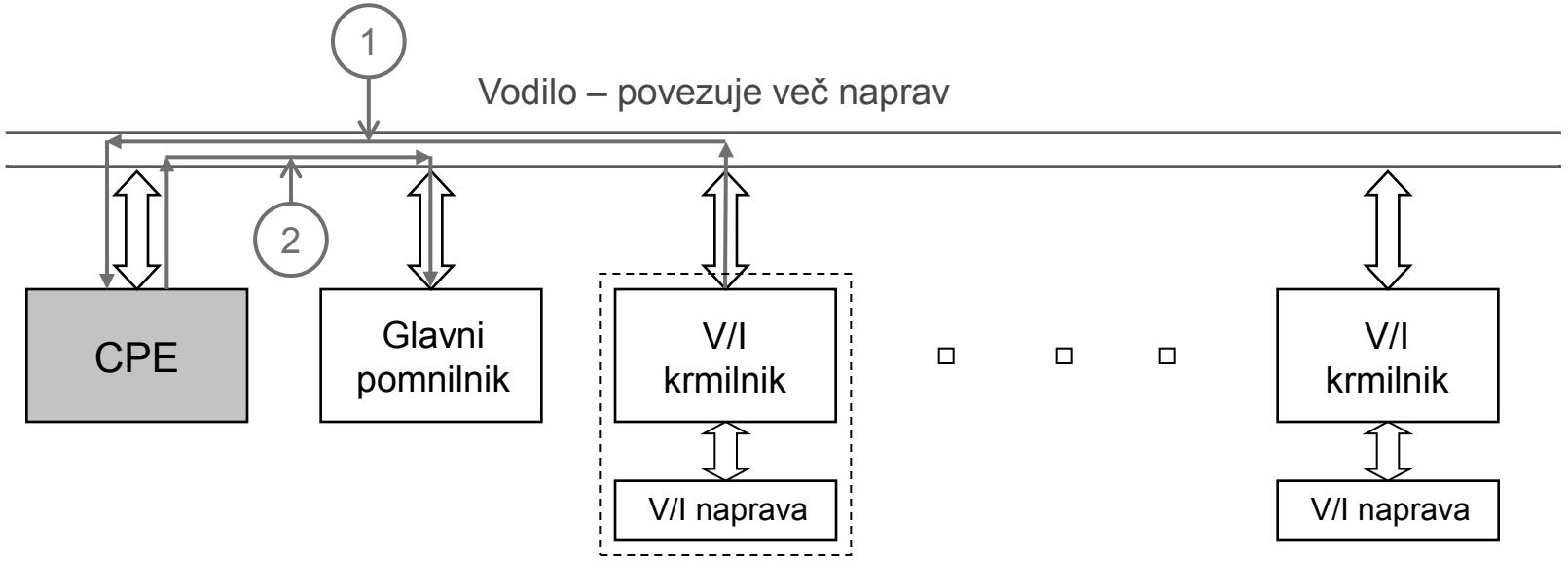
- število brez predznaka: 139 (desetiško)
- število s predznakom: - 11 (desetiško)
- znak v razširjeni ASCII abecedi: <
- strojni ukaz (op. koda 68HC11): ADDA
- pomnilniški naslov: 139 (desetiško)
- kombinacijo bitov

3.4 Vhod in izhod v von Neumannovem računalniku

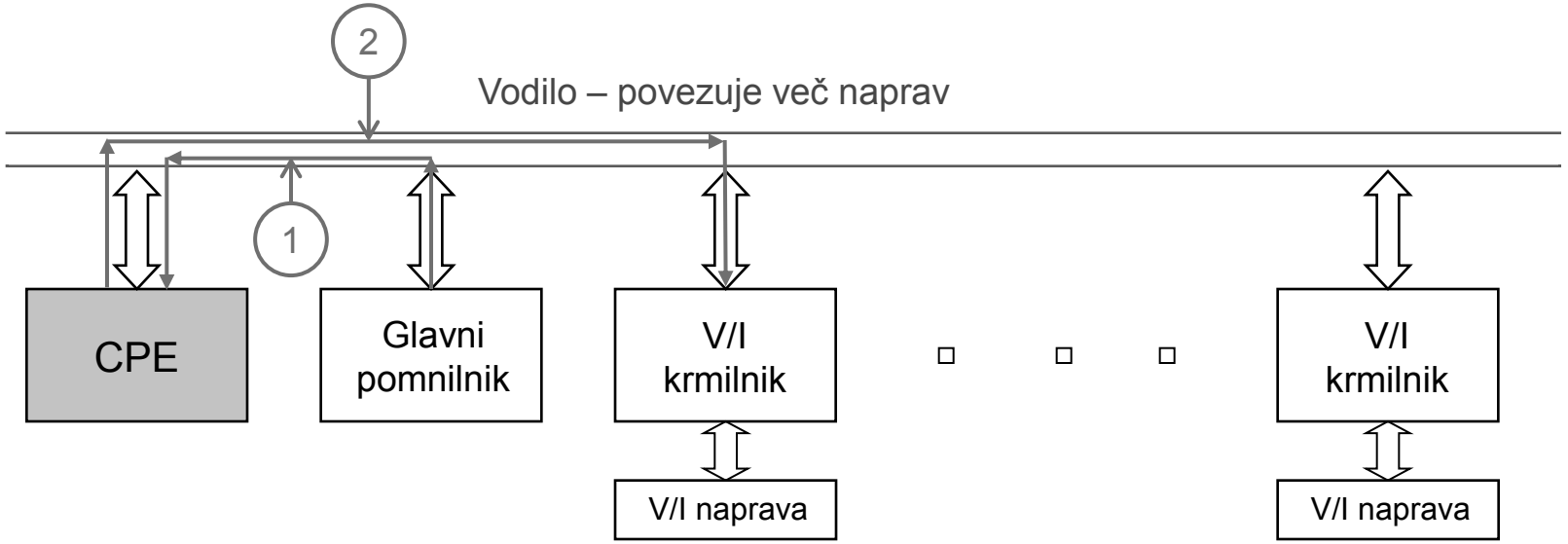
- V/I naprave, ki služijo pretvarjanju informacij iz ene oblike v drugo (tipkovnica, miška, zaslon, tiskalnik . . .)
- V/I naprave za shranjevanje informacij – pomožni pomnilniki (trdi disk, SSD, magnetni trak, DVD . . .)
- Osnovni način delovanja V/I sistema: prenos podatkov med V/I sistemom in glavnim pomnilnikom

Izvedbe V/I prenosa

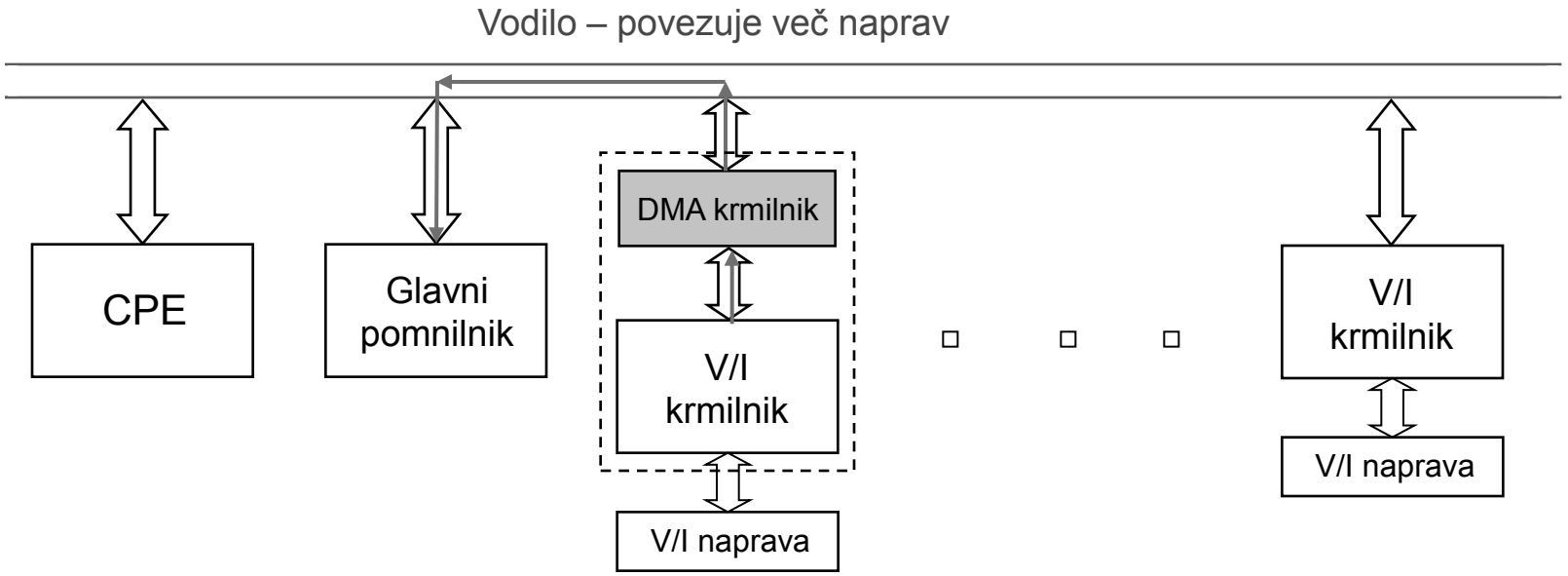
- Programski vhod/izhod (programmed I/O – PIO)
  - Z V/I napravo komunicira CPE.
  - Potek prenosa
    - V/I → CPE → pomnilnik (branje iz V/I naprave)



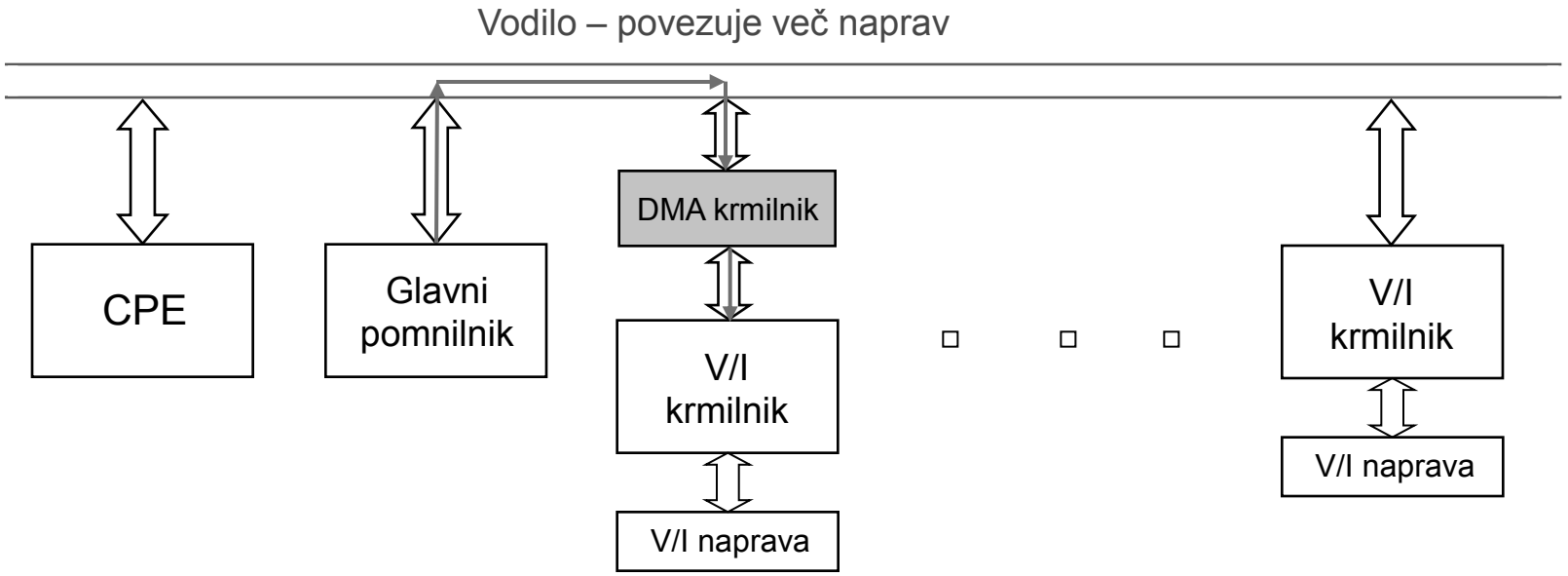
- Programski vhod/izhod (programmed I/O – PIO)
  - Z V/I napravo komunicira CPE.
  - Potek prenosa
    - Pomnilnik → CPE → V/I naprava (pisanje v V/I napravo)



- Neposredni dostop do pomnilnika (direct memory access – DMA)
  - Potreben je DMA krmilnik.
  - Potek prenosa
    - V/I → DMA krmilnik → pomnilnik

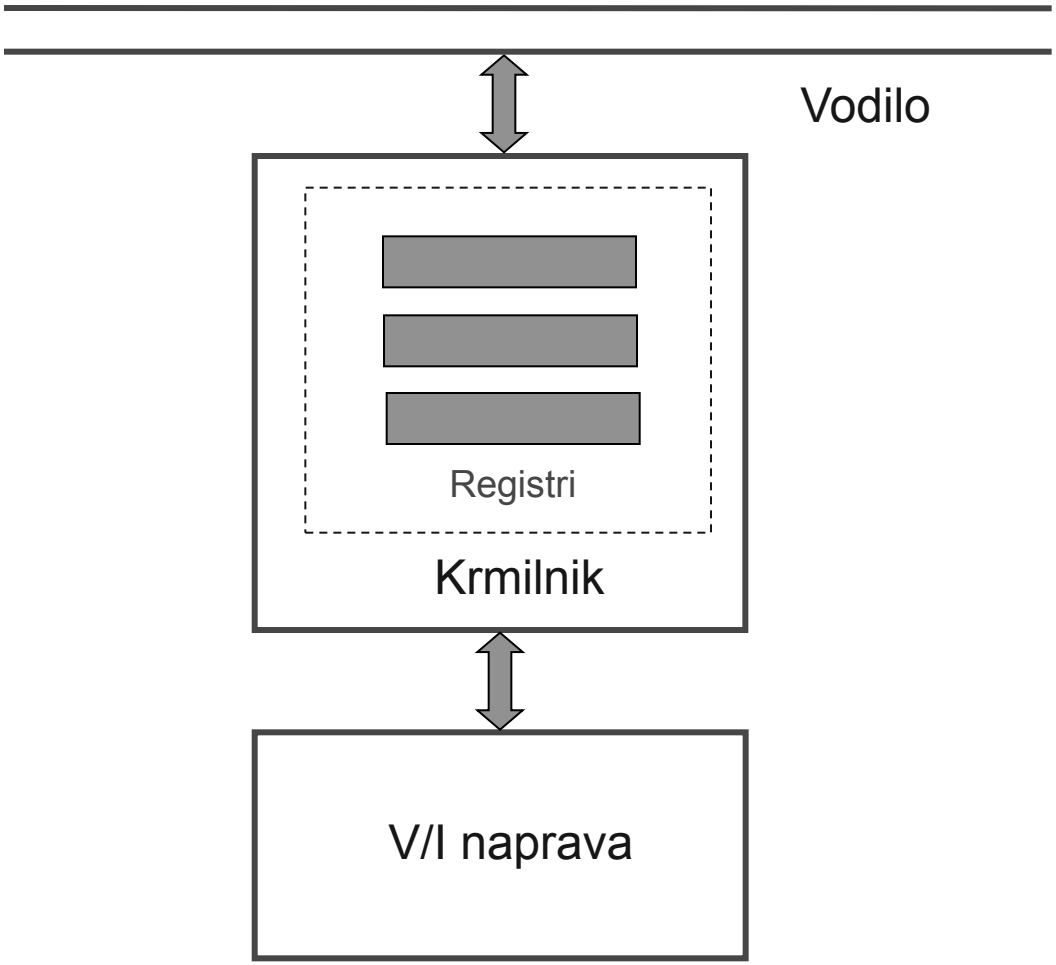


- Neposredni dostop do pomnilnika (direct memory access – DMA)
  - Potreben je DMA krmilnik.
  - Potek prenosa
    - Pomnilnik → DMA krmilnik →V/I naprava



Krmilnik V/I naprave

- Vsaka V/I naprava je priključena preko krmilnika naprave.
- Gledano iz CPE je krmilnik videti kot določeno število registrov, v katere se lahko piše ali iz njih bere.
- Pisanje (lahko tudi branje) v določen register (ukazni register) lahko sproži operacijo v V/I napravi (ukaz napravi).
- Branje iz določenega registra (statusni register) odraža stanje naprave po V/I operaciji (status naprave).
- Tudi prenos informacij poteka z branjem iz določenega registra krmilnika ali s pisanjem v register krmilnika (podatkovni register).



Naslovi registrov v V/I krmilniku

- **Pomnilniško preslikani vhod/izhod** (memory mapped I/O) – registri krmilnikov so iz CPE videti enako kot pomnilniške lokacije. Naslovi registrov zasedajo pomnilniški naslovni prostor.

- **Ločeni vhodno/izhodni naslovni prostor** – potrebni so posebni vhodno/izhodni ukazi za dostop do registrov krmilnikov. Register v CPE in pomnilniška beseda imata lahko enak naslov.

- **Posredno naslavljanje preko V/I procesorjev** – naslovi registrov krmilnikov so v posebnem naslovnem prostoru, do katerega imajo dostop samo V/I procesorji.

Prenosne poti v von Neumannovem računalniku

- Tvorijo povezavo med CPE, glavnim pomnilnikom in V/I sistemom.
- Hiter računalnik zahteva hitre povezave, ki omogočajo prenos čim več podatkov v časovni enoti.

- Ena prenosna pot omogoča, da se v nekem trenutku izvaja samo en prenos.

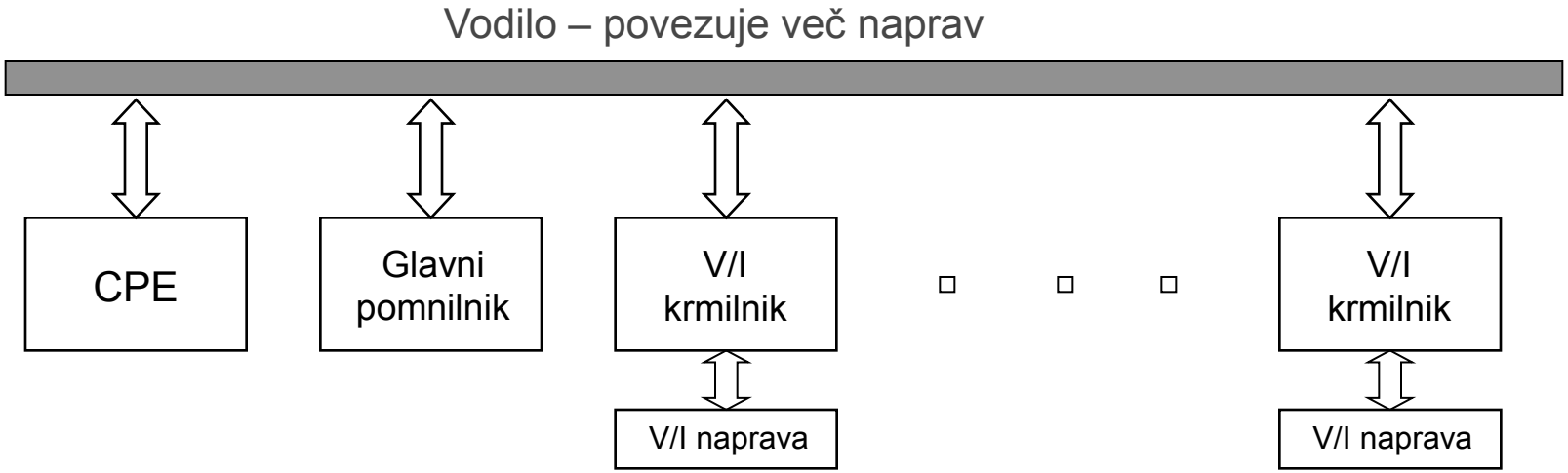
- Količino informacije, ki jo lahko prenesemo po določeni poti, lahko povečamo:

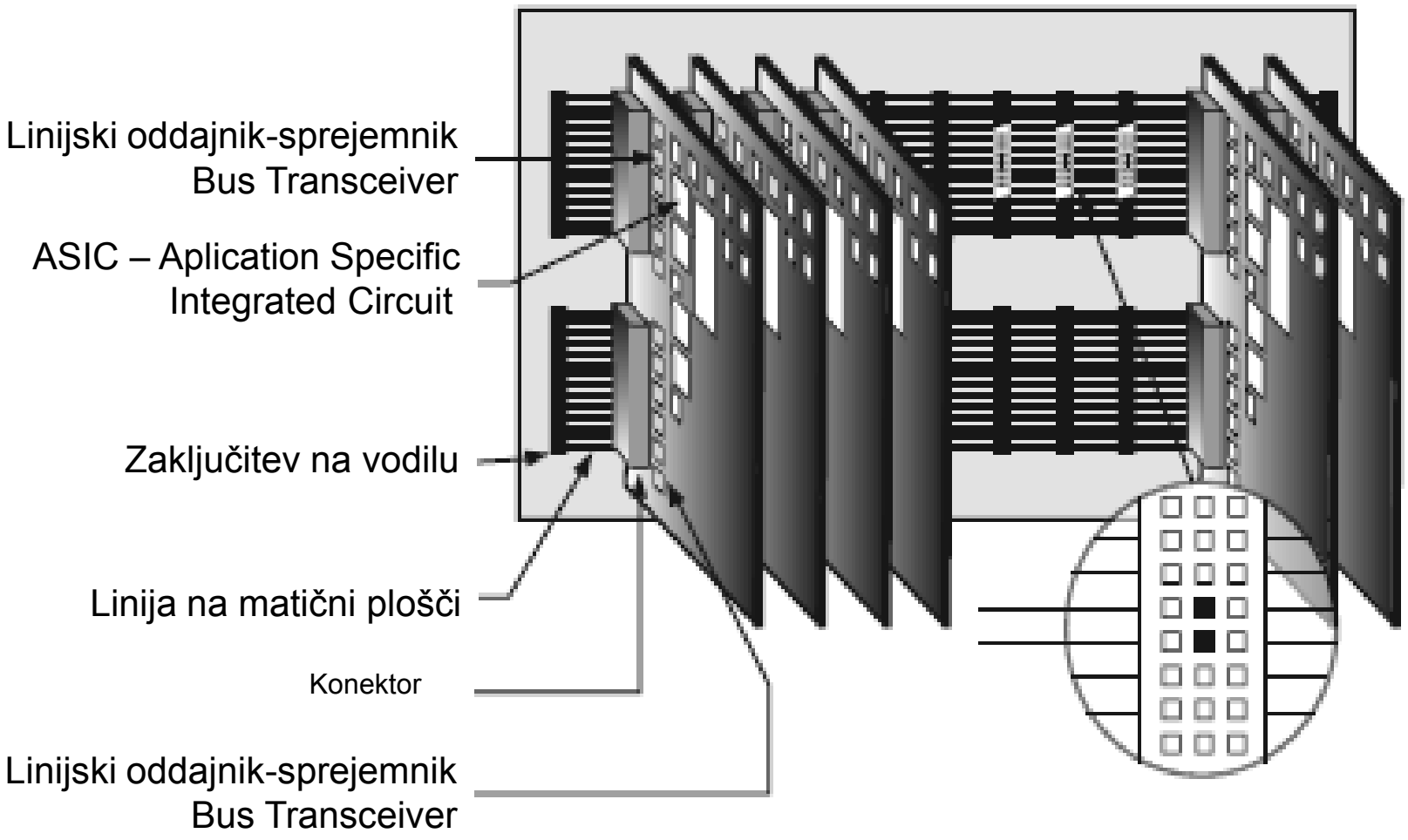
  - s krajšanjem časa, ki je potreben za en prenos,

  - z večanjem števila bitov, ki se prenesejo v enem prenosu (= s povečanjem širine prenosne poti).

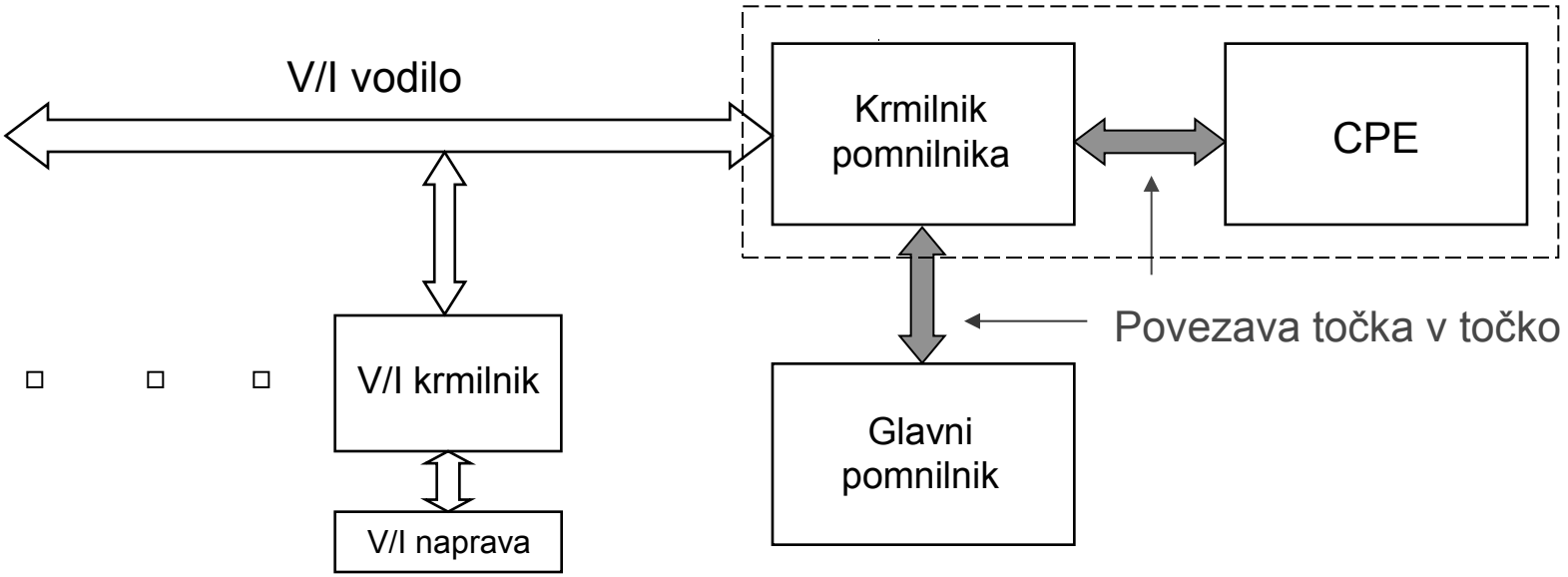
- Na hitrost vpliva zato tudi število prenosnih poti – govorimo o povezovalnih strukturah.
- Vrste prenosnih poti:
  - vodilo (bus)
  - povezava točka v točko (point to point)

- Vodilo (angl. bus)
  - Vodilo si delijo vse enote, ki so priključene nanj.
  - Fizično je vodilo množica paralelnih linij (žic), po katerih potujejo električni signali.
  - Linije imajo odcepe, na katere so priključene enote.





- Povezava točka v točko (angl. point to point)
  - Prenosna pot, ki povezuje samo dve napravi



Signali, ki se prenašajo po prenosnih poteh

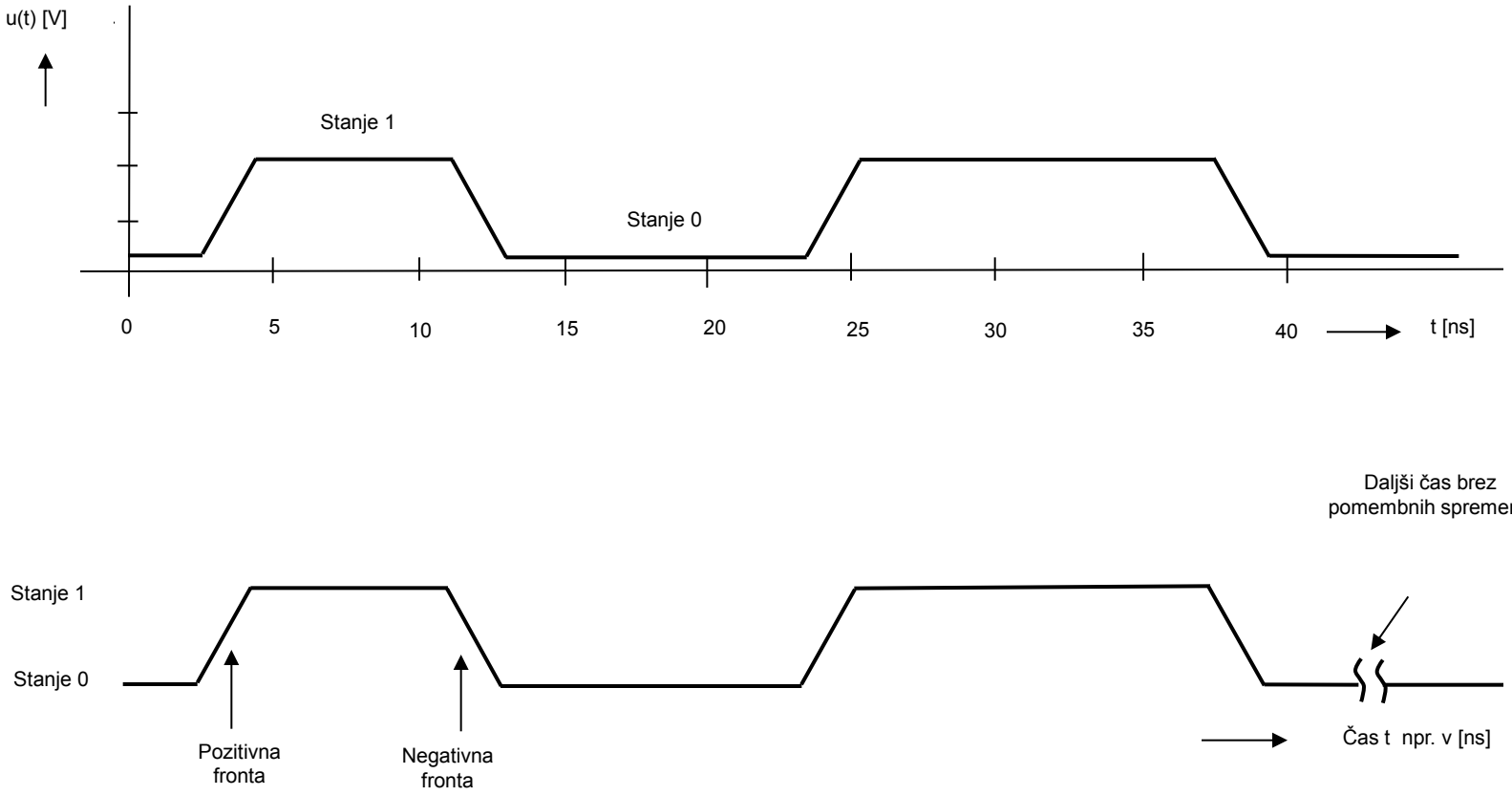
- Po linijah, ki sestavljajo prenosno pot, se prenašajo tri vrste signalov:
  - naslovni signali
  - kontrolni signali
  - podatkovni signali

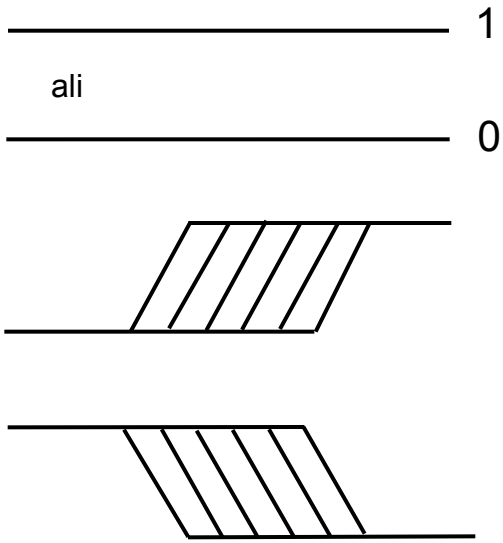
- Naslovni signali
  - Določajo naslov pomnilniške besede ali V/I naprave (registra v krmilniku V/I naprave), na katero se nanaša prenos.
  - Število naslovnih signalov (bitov) določa velikost naslovnega prostora.
  - Običajne oznake npr. pri 32-bitnih naslovih A0 – A31

- Kontrolni signali
  - Določajo smer prenosa (branje ali pisanje), število prenesenih bitov in časovno zaporedje dogodkov pri prenosu itn.
  - Nekateri kontrolni signali:
    - urin signal (običajna oznaka signala CLK),
    - bralno-pisalni signal (oznaka R/W),
    - signal za resetiranje procesorja (oznaka RESET)
    - prekinitvena zahteva (oznaka INT ali IRQ),
    - . . .

- Podatkovni signali
  - Število podatkovnih signalov (linij) je enako številu bitov, ki se naenkrat prenašajo po prenosni poti – širina prenosne poti.
  - Pri širini 64 bitov so možni 8-, 16-, 32- in 64-bitni prenosi.
  - Običajne oznake npr. pri 64-bitni širini D0 – D63

Predstavitev signalov s časovnimi diagrami

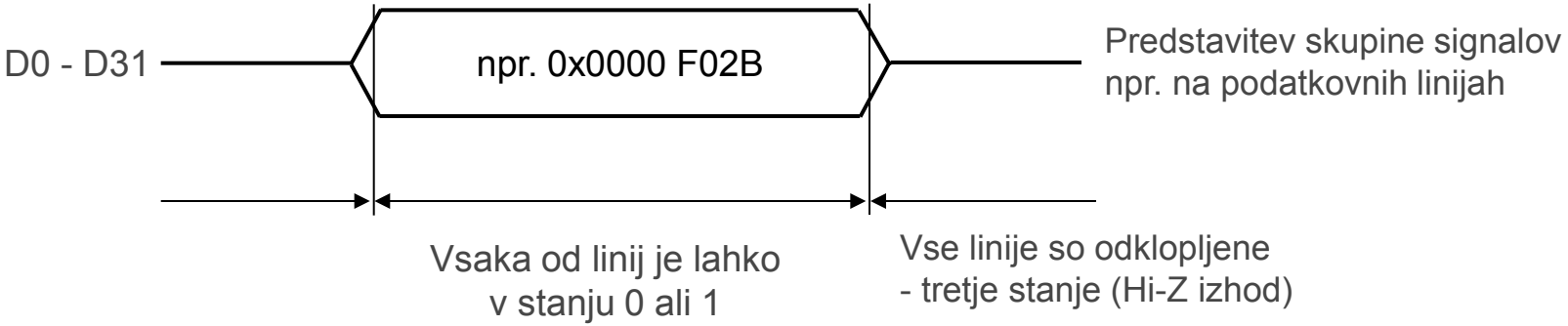




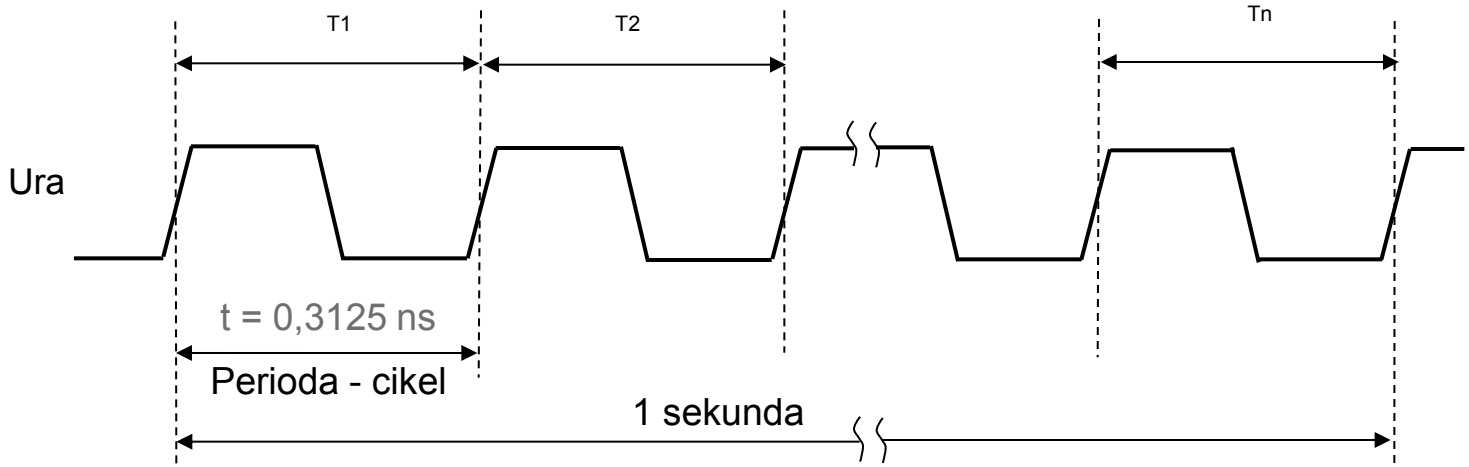
Signal je stabilen.

Signal se spreminja iz stanja 0 v stanje 1.

Signal se spreminja iz stanja 1 v stanje 0.



Urin signal - periodični pravokotni signal



Pri f = 3,2 GHz je v 1 sekundi 3.200.000.000 period

Frekvenca periodičnega signala f = število period (ciklov) v 1 sekundi

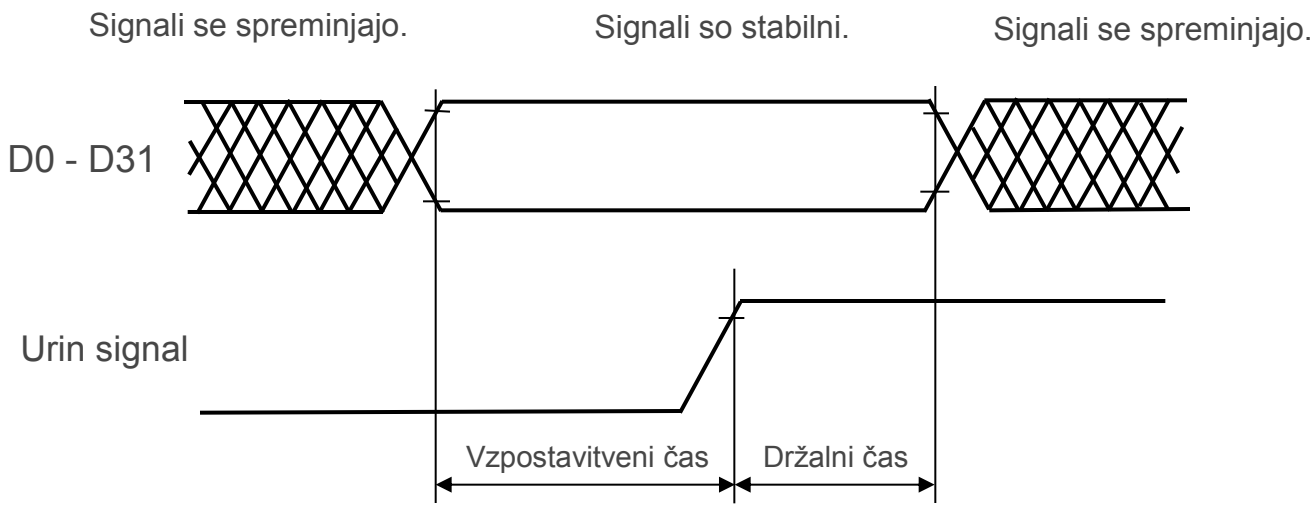
Enota za frekvenco je Hertz (Hz): 1 Hz = 1 perioda/s = 1 s^{-1}

Čas trajanja ene periode t = 1 / f

Primer: f = 3,2 GHz

$$t = \frac{1}{f} = \frac{1}{3{,}2 \cdot GHz} = \frac{1}{3{,}2 \cdot 10^9 \frac{1}{s}} = 0{,}3125 \cdot 10^{-9}\, s = 0{,}3125 ns$$

Uporaba urinega signala



Vloge naprav pri prenosu

- Vsak prenos poteka samo med dvema napravama.
- Vlogi naprav pri prenosu
  - gospodar (master) – vodi prenos
  - suženj (slave)

- Pri vodilih je potencialnih gospodarjev lahko več – pri prenosu je aktiven samo eden.
- Pri povezavi točka v točko je samo en gospodar.
- Pri več gospodarjih na vodilu (drugi procesor, DMA krmilnik itn.) je potrebna arbitraža.
- Arbitraža - mehanizem za dogovor, kdo od gospodarjev bo imel pri prenosu nadzor nad vodilom.

Zaporedje dogodkov pri prenosu

- Naslov
  - Gospodar pošlje na prenosno pot naslov, do katerega želi dostop (naslovne signale postavi v stanje, ki ustreza ciljnemu naslovu).
- Kontrolni signali
  - Gospodar poda smer prenosa:
    - bralni prenos
    - pisalni prenos
  - Število bitov, ki naj se prenesejo (če so možni prenosi različnih širin)

- Podatki
  - Kadar je smer prenosa branje, gospodar pričakuje podatek iz prenosne poti (podatkovne signale).
  - Kadar je smer prenosa pisanje, gospodar na prenosno pot pošlje tudi podatek (podatkovne signale).
- Začetek in konec prenosa
  - Gospodar s kontrolnimi signali tudi določi začetek in konec prenosa.

- Vse enote, priključene na prenosno pot (sužnji), primerjajo svoje naslove z naslovom na naslovnih linijah.

- Enota (suženj), ki ugotovi enakost, opravi prenos, ki ga zahteva gospodar.

- **! Prenos:** enkraten paralelni prenos največ toliko bitov, kot je širina podatkovne prenosne poti.

Kapaciteta prenosne poti

- Kapaciteta prenosne poti B (Bandwith, Throughput) je največje možno število:
  - prenosov (Transfers) v sekundi (T/s)
    (MT/s = 10^6 T/s, GT/s = 10^9 T/s)
  - prenesenih bitov v sekundi (b/s, Mb/s, Gb/s)
  - prenesenih bajtov v sekundi (B/s, MB/s, GB/s)
    - MB/s = 10^6 B/s; GB/s = 10^9 B/s

$$B = f_{vodila} \cdot \frac{1}{Štev\ period\ za\ en\ prenos} \cdot Širina\ vodila$$

- *B* = kapaciteta (zmogljivost) prenosne poti v bitih/sekundo
  (ali v bajtih/sekundo, če je širina v bajtih)
- f_{vodila} = frekvenca urinega signala na vodilu v [Hz] = [1/s]
  (= število urinih period v 1 sekundi)
- *Število period za en prenos* = trajanje enega prenosa v urinih periodah
- *Širina vodila* = število prenesenih bitov (ali bajtov) v enem prenosu

- Primer: Kapaciteta PCI vodila
  - Širina prenosne poti = 32 bitov (= 4 bajte = 4 B)
  - Frekvenca urinega signala na vodilu f_{vodila}= 33 MHz
  - (33MHz = 33*10^6 Hz = 33*10^6 s^{-1} !)
  - En prenos traja eno urino periodo.

$$B\,[b/s] = 33\cdot 10^6[1/s]\cdot 1\,[perioda\ /\ prenos]\cdot 32[bitov] = 1056\cdot 10^6[b/s]$$

$$B\,[B/s] = \frac{1056\cdot 10^6[b/s]}{8[b/B]} = 132\cdot 10^6[B/s] = 132[MB/s]$$

  - ali

$$B\,[B/s] = 33\cdot 10^6[1/s]\cdot 1\,[perioda\ /\ prenos]\cdot 4[B] =$$
$$= 132\cdot 10^6[B/s] = 132[MB/s]$$

Vrsta prenosne poti	Kapaciteta v bitih/s	Kapaciteta v Bajtih/s
ISA (8 bitov, 4,77 MHz)	9,6 Mb/s	1,2 MB/s
PCI (32 bitov, 33 MHz)	1056 Mb/s	133 MB/s
PCI (64 bitov, 66 MHz)	4224 Mb/s	528 MB/s
PCI Express 2.0 (x16)	64 Gb/s	8 GB/s
HT (AMD 3.2 GHz)	409,6 Gb/s	51,2 GB/s
QPI (Intel 3,2 GHz)	204,8 Gb/s	25,6 GB/s

3.5 Prekinitve in pasti

- **Prekinitev** (interrupt) je dogodek, ki povzroči, da CPE začasno preneha izvajati trenutni program in začne izvajati neki drugi program - prekinitveni servisni program (PSP).

- **Prekinitveni servisni program** je kratek program, tudi njegov čas izvajanja je običajno omejen, da ne moti izvajanja programa, ki je bil prekinjen.

- **Zahteva za prekinitev** (Interrupt request) je signal, ki pride v CPE od zunaj:
  - od neke V/I naprave,
  - od nekega drugega dela računalnika.
- S pomočjo prekinitev lahko drugi deli računalnika ali zunanje naprave pridejo do uslug CPE.
- Prekinitev lahko smatramo tudi kot zahtevo za servisiranje. Servisiranje izvede PSP (prekinitveni servisni program).

- **Past (Trap)** je posebna vrsta prekinitev, ki jo zahteva CPE ob nekem posebnem dogodku ali na zahtevo programerja (poseben strojni ukaz, npr. SWI).

- Gledano iz CPE je izvor pasti znotraj CPE, izvor prekinitev pa zunaj CPE.

- Odziv CPE na prekinitve in pasti pa je enak.

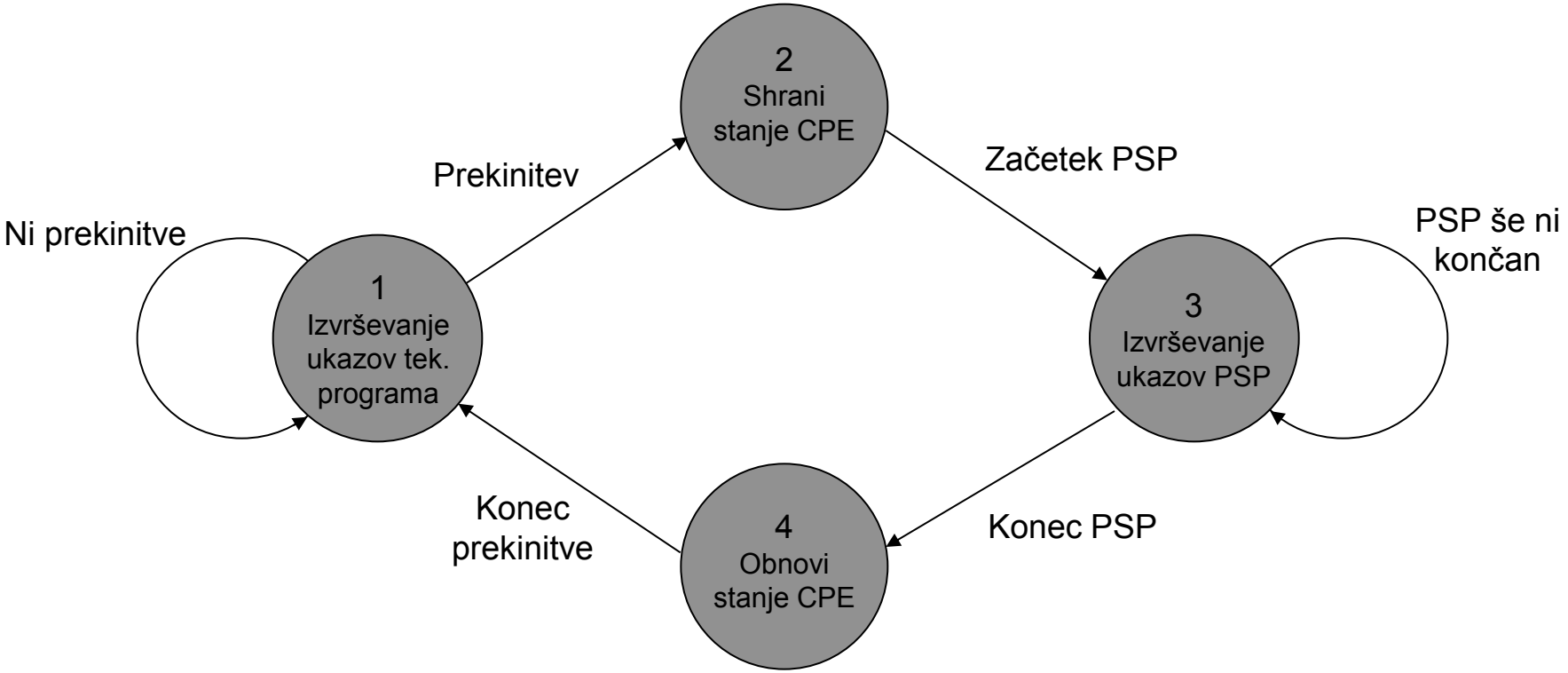
- Nekateri primeri uporabe prekinitev in pasti
  - Prekinitve
    - Zahteve V/I naprav ob različnih dogodkih (npr. zaključek V/I operacije, V/I naprava lahko sprejme ali pošlje nov podatek)
    - Signal iz senzorja (npr. prenizka napetost baterijskega napajanja)
    -
  - Pasti
    - Napake pri delovanju računalnika
    - Aritmetični preliv
    - Dostop do zaščitene pomnilniške besede
    - Uporaba nedefiniranega ukaza
    - Prehod v bolj privilegiran način delovanja
    - Napaka strani pri navideznem pomnilniku
    -

Dogajanje pri prekinitvah

- Pri prekinitvah je osnovna zahteva, da so “nevidne” (transparentne).

- Dogajanje pri prekinitvah lahko opišemo z diagramom s štirimi stanji:

  - 1. Izvrševanje ukazov tekočega programa
  - 2. Shranjevanje stanja CPE ob zahtevi za prekinitev
  - 3. Skok na PSP in izvrševanje njegovih ukazov (tudi ta program je lahko prekinjen)
  - 4. Vrnitev iz PSP in obnovitev stanja CPE, ki mora biti enako kot na začetku točke 2

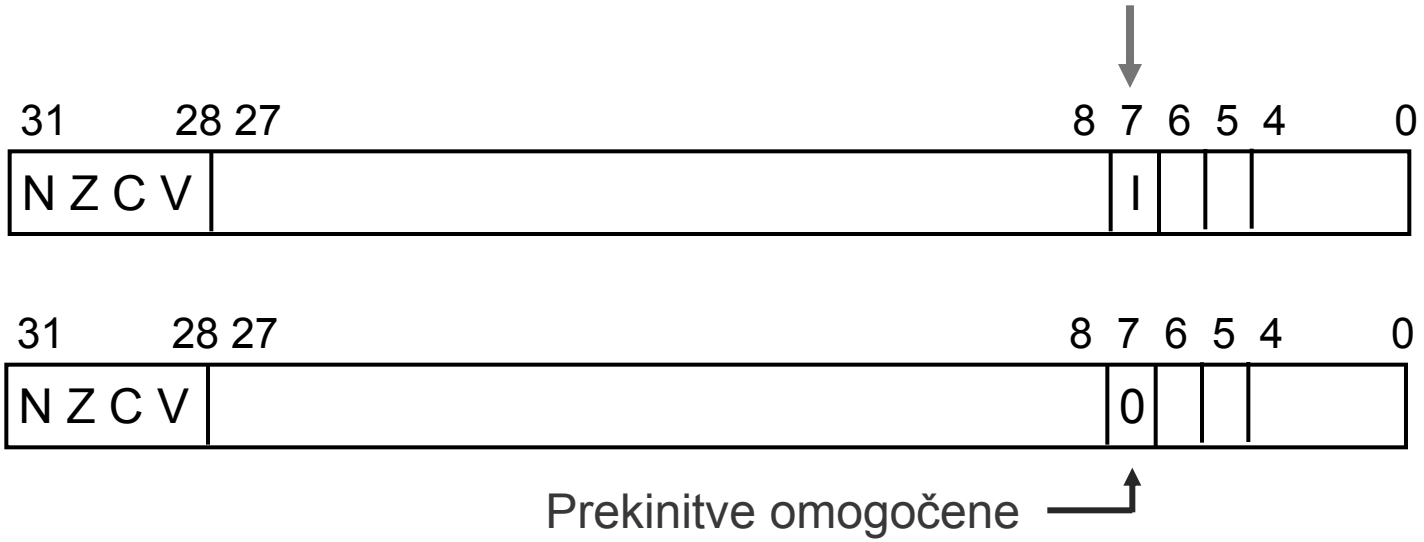

2
Shrani
stanje CPE
Prekinitev
Začetek PSP
Ni prekinitve
PSP še ni
končan
1
Izvrševanje
ukazov tek.
programa
3
Izvrševanje
ukazov PSP
Konec
prekinitve
Konec PSP
4
Obnovi
stanje CPE

- CPE s prekinitvenimi sposobnostmi mora biti zgrajena tako, da je na neki način rešenih pet problemov:
  - Kdaj CPE reagira na prekinitveno zahtevo
  - Kako je zagotovljena nevidnost prekinitev
  - Kje CPE dobi naslov prekinitveno-servisnega programa
  - Prioriteta prekinitvenih zahtev
  - Potrjevanje prekinitvene zahteve

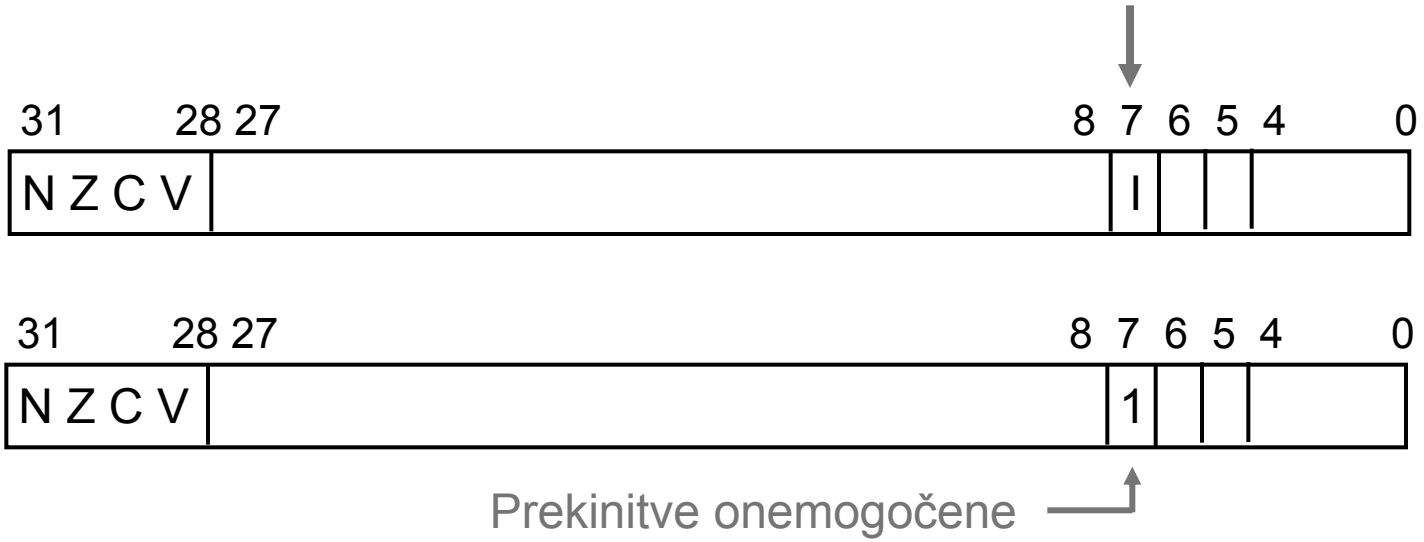
Kdaj CPE reagira na prekinitveno zahtevo

- Najenostavnejša rešitev: po koncu izvrševanja tekočega ukaza (pred prevzemom naslednjega).
  - Nevidnost prekinitev je v tem primeru zagotovljena tako, da se ohrani samo vsebina programsko dostopnih registrov (vseh ali samo nekaterih – tistih, ki jih uporablja PSP).
- Če CPE reagira na prekinitvenih zahtevo v poljubnem stanju, je treba ohraniti vsebino vseh registrov - programsko dostopnih in programsko nedostopnih.
  - Uporaba pri pasteh, če je vzrok tak, da dokončanje ukaza ni mogoče.

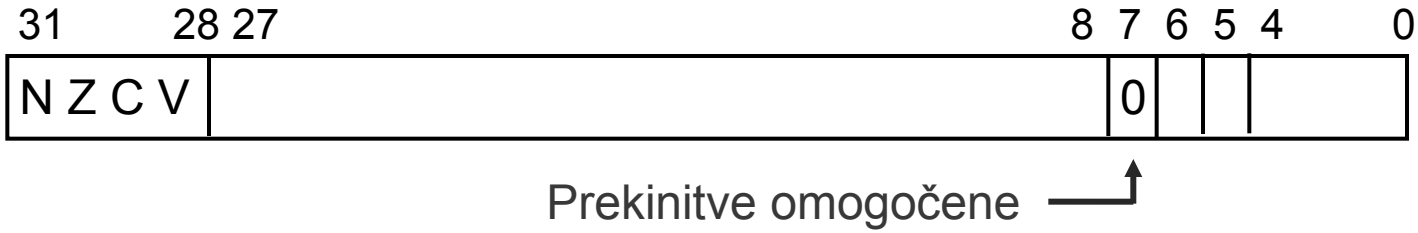
- CPE mora programerju omogočati nadzor nad prekinitvami - maskiranje prekinitev. Primer register CPSR (Current Program Status Register) v procesorju ARM:
  - I - bit (bit 7) v registru CPSR procesorja ARM
  - I ← 1 prekinitve onemogočene
  - I ← 0 prekinitve omogočene



- CPE mora programerju omogočati nadzor nad prekinitvami - maskiranje prekinitev. Primer register CPSR (Current Program Status Register) v procesorju ARM:
  - I - bit (bit 7) v registru CPSR procesorja ARM
  - I ← 1 prekinitve onemogočene
  - I ← 0 prekinitve omogočene



- Omogočanje prekinitev pri procesorju ARM (vpis 0 v bit 7 CPSR)




```
mrs  r0, cpsr       @ R0 ← CPSR
bic  r0, r0, #0x80  @ (b7←0) prek. omogočene

msr  cpsr, r0       @ CPSR ← R0
```


- Procesor prekinitve avtomatsko onemogoči:
  - po vklopu računalnika (ob RESET-u), da se lahko izvede inicializacija V/I naprav, sklada itn.,
  - po sprejetju prekinitvene zahteve, da ne bi bila sprejeta nova prekinitev, preden je shranjeno stanje CPE.
- Prekinitve ponovno omogoči programer v programu:
  - ko je končana inicializacija po vklopu,
  - po sprejeti prekinitvi v prekinitvenem servisnem programu (PSP), če presodi, da CPE lahko sprejema vgnezdene prekinitve.

Kako je zagotovljena nevidnost prekinitev

- Po koncu prekinitvenega servisnega programa se mora prekinjeni program nadaljevati, kot da prekinitve ni bilo.
- Prekinjeni program ne bo “čutil” posledic prekinitve, če bo stanje CPE (stanje vseh registrov) ob vrnitvi vanj enako kot pri vstopu v PSP.
- CPE mora ob prekinitvi shraniti najmanj stanje PC-ja, saj drugače ni možna vrnitev iz PSP v prekinjeni program.

- Pri računalnikih z malo registri je CPE večkrat narejena tako, da ob prekinitvi avtomatsko shrani vse registre.
- Pri računalnikih z veliko registri ta rešitev zaradi počasnosti ni primerna.
- Najboljša rešitev je, da PSP (programer, ki napiše PSP) sam poskrbi za shranjevanje in obnavljanje tistih registrov, katerih vsebino spreminja.
- Stanje CPE ob prekinitvi se lahko shrani v posebne registre v CPE ali pa v sklad v pomnilniku.

Kje CPE dobi naslov PSP

- Pasti imajo izvor v CPE, zato tega problema ni, ker CPE tvori tudi naslov PSP-ja, ki obdela past.
- Prekinitve prihajajo v CPE od zunaj, zato CPE potrebuje informacijo, kateri PSP naj se izvede.
- Ta informacija pove, katera naprava je zahtevala prekinitev in s tem kateri PSP naj se izvede.
- Temu pravimo tudi **prepoznavanje naprave**.

- Če ima procesor več prekinitvenih vhodov in je na vsakega priključena ena naprava (izvor prek. zahteve), dodatne informacije za razpoznavanje niso potrebne.

- Če je na en prekinitveni vhod priključenih več naprav, je potreben dodaten mehanizem za prepoznavanje naprave ali vzroka prekinitvene zahteve.
  - Programsko izpraševanje (programska rešitev)
  - Vektorske prekinitve (strojna rešitev)

- Programsko izpraševanje (polling)
  - V CPE je vgrajen fiksni naslov, na katerem CPE prebere naslov PSP-ja.
  - Vsaka V/I naprava ima v enem od svojih registrov bit, ki pove, ali naprava zahteva prekinitev.
  - Prekinitveni servisni program v določenem vrstnem redu ($\Rightarrow$ prioriteta) bere registre V/I naprav.
  - Ko naleti na prvo napravo, ki je zahtevala prekinitev, izvrši skok na njen PSP.
  - Vrstni red branja registrov V/I naprav določa prioriteto obravnave prekinitvenih zahtev.

- Vektorske prekinitve
  - Ko CPE sprejme prekinitveno zahtevo, pošlje na vodilo potrditveni signal (INTA - interrupt acknowledge).
  - Krmilnik V/I naprave, ki je zahtevala prekinitev, pošlje na vodilo (na podatkovne linije) številko, ki ji rečemo **prekinitveni vektor.**
  - CPE prekinitveni vektor uporabi kot kazalec do tabele v pomnilniku z naslovi prekinitveno-servisnih programov (PSP).
  - Naslov iz tabele se vpiše v programski števec in tako PC kaže na prvi ukaz prekinitveno-servisnega programa naprave, ki je zahtevala prekinitev.

Prioriteta prekinitvenih zahtev

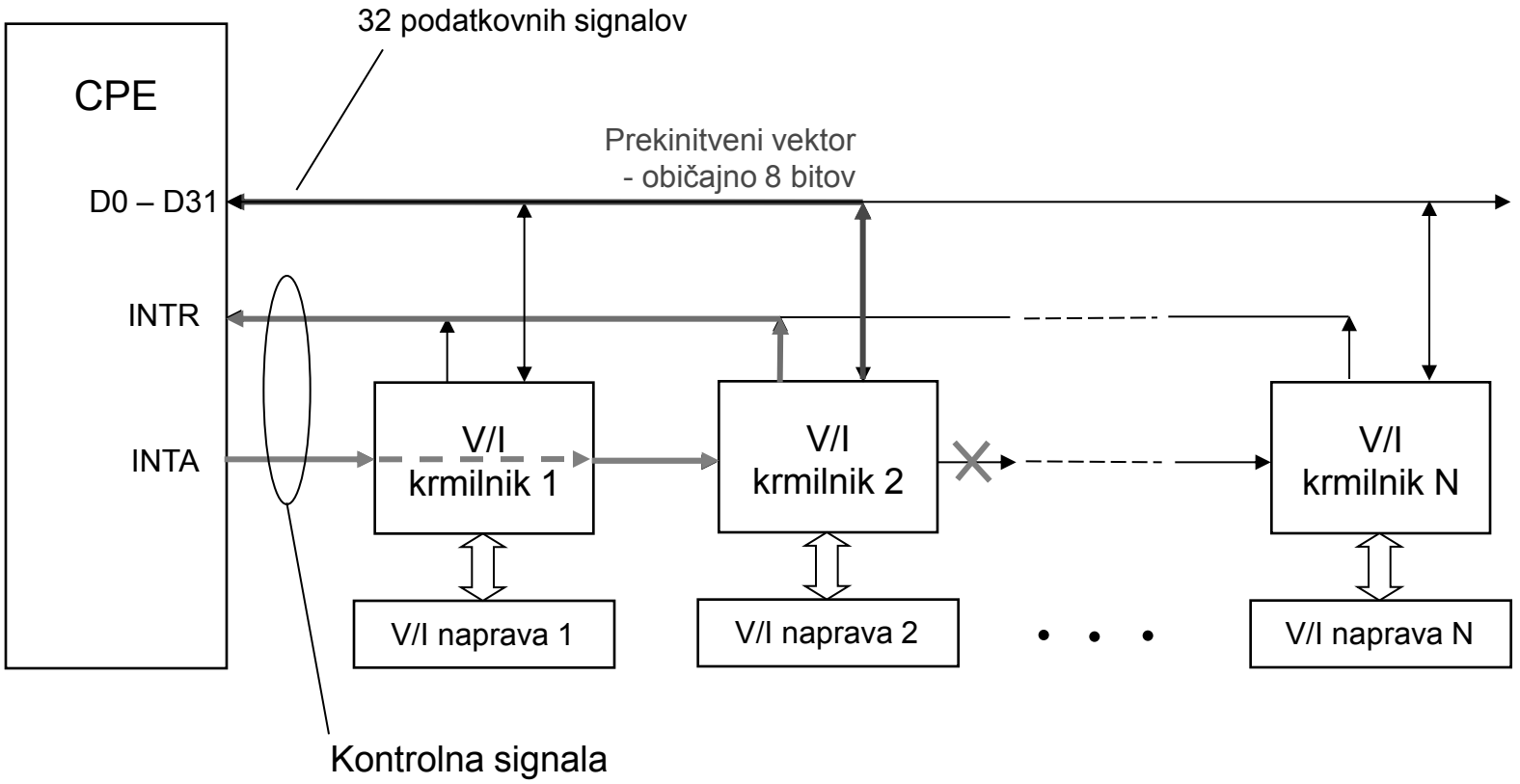
- Če je na prekinitveni vhod CPE priključenih več naprav ali ima CPE več prekinitvenih vhodov, je treba določiti prioriteto prekinitvenih zahtev.
- Če je hkrati prisotnih več prekinitvenih zahtev, prioriteta določa, katera zahteva se bo upoštevala prva.
- V mnogih računalnikih lahko prekinitve z višjo prioriteto prekinejo prekinitvene servisne programe z nižjo prioriteto - vgnezdene prekinitve.

- Če je na en prekinitveni vhod priključenih več naprav, se prioriteta lahko določa:
  - s programskim izpraševanjem (polling) (programska rešitev) ⇒
    ⇒ glej str. 89,
  - z marjetično verigo (daisy chain) (strojna rešitev).

- Marjetična veriga
  - Marjetična veriga se pogosto uporablja za določanje prioritete pri vektorskih prekinitvah.
  - CPE kot odgovor na prekinitveno zahtevo v prekinitveno prevzemnem ciklu aktivira signal (npr. INTA – Interrupt Acknowledge) in ga pošlje v naprave na tem prekinitvenem vhodu.

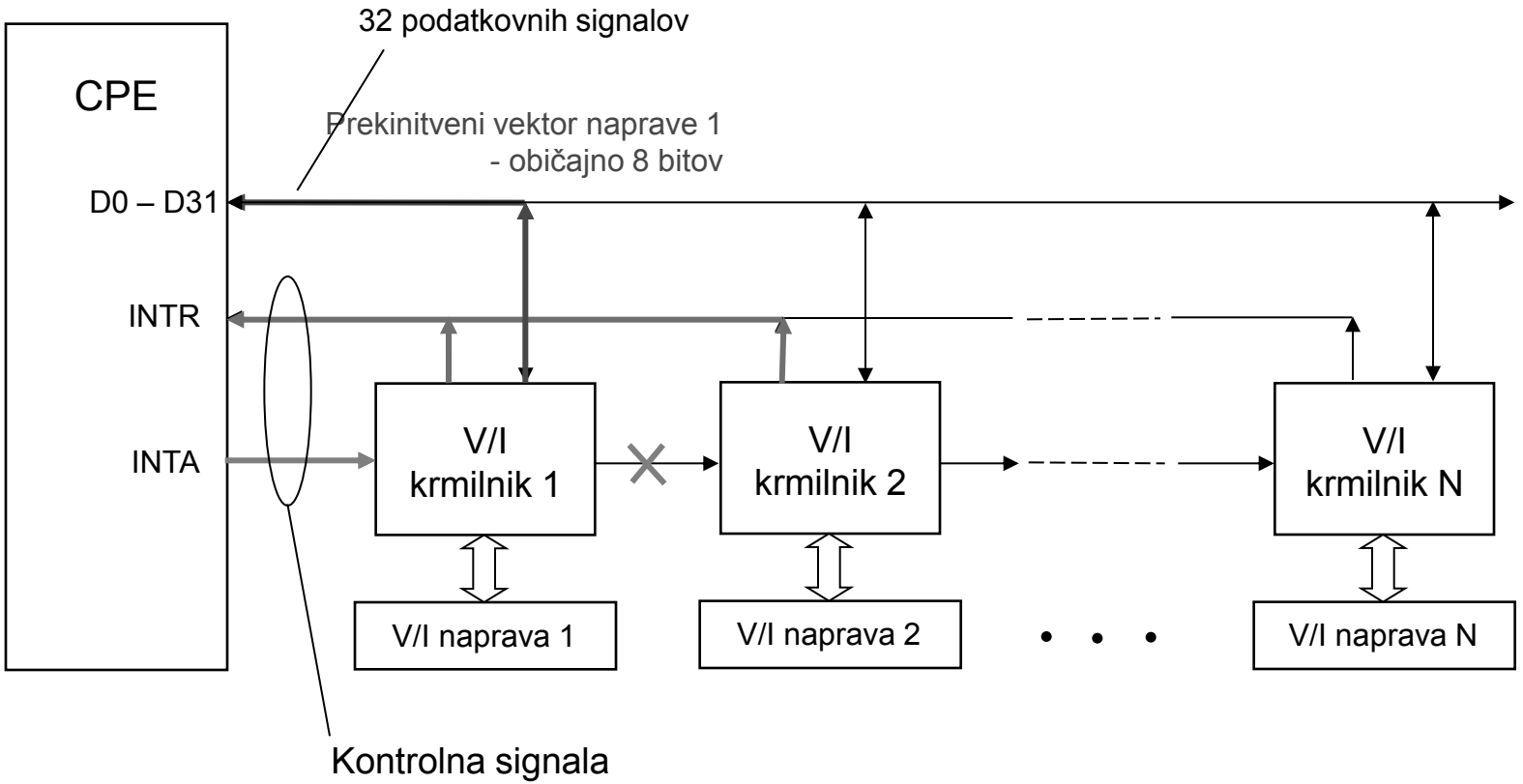
Marjetična veriga - prekinitev zahteva naprava 2

INTR (Interrupt Request – signal za prekinitveno zahtevo
INTA (Interrupt Acknowledge) – signal za potrditev prekinitve



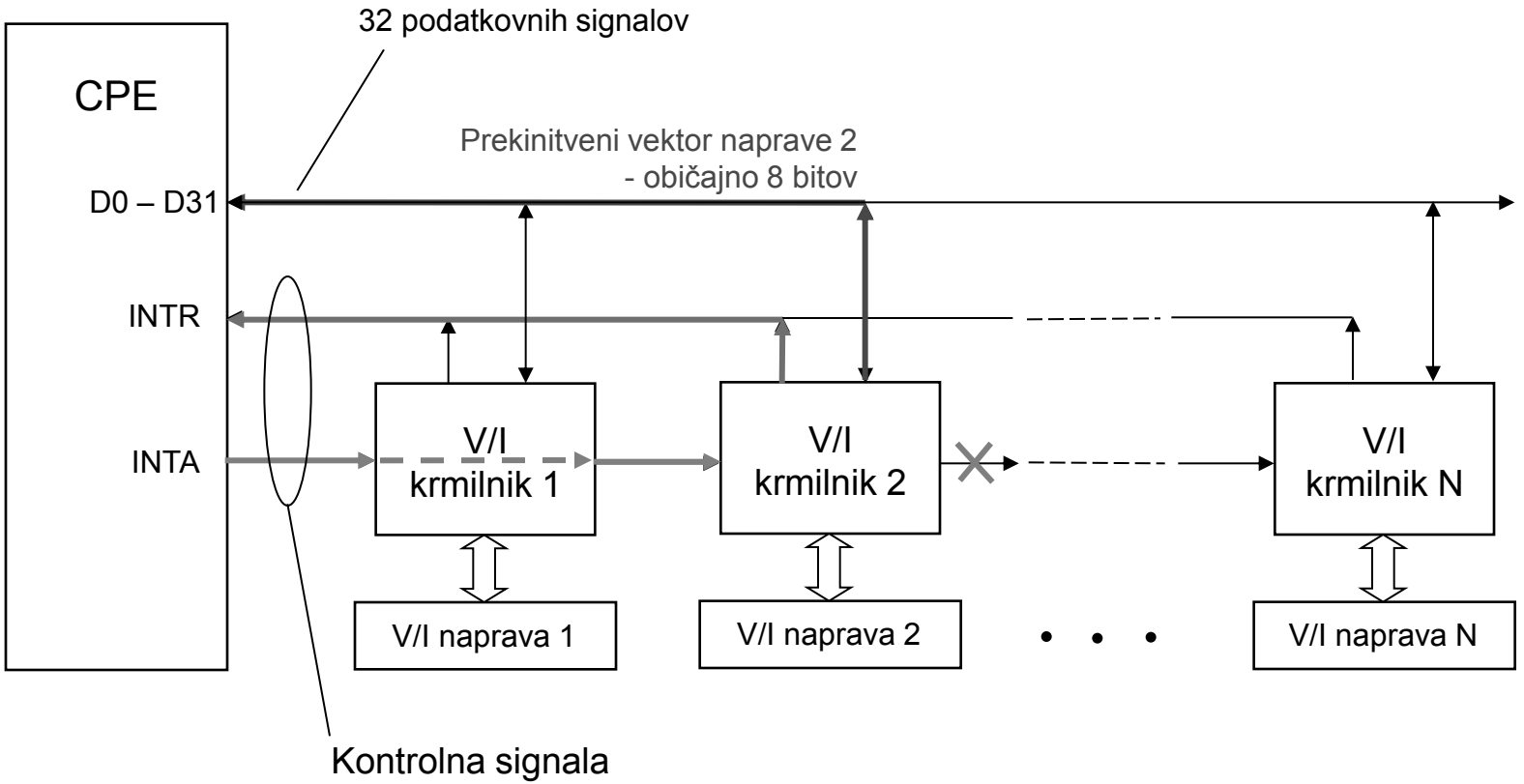
Marjetična veriga - prekinitev zahtevata istočasno napravi 1 in 2

INTR – signal za prekinitveno zahtevo
INTA – signal za potrditev prekinitve



Marjetična veriga - prekinitev zahtevata istočasno napravi 1 in 2

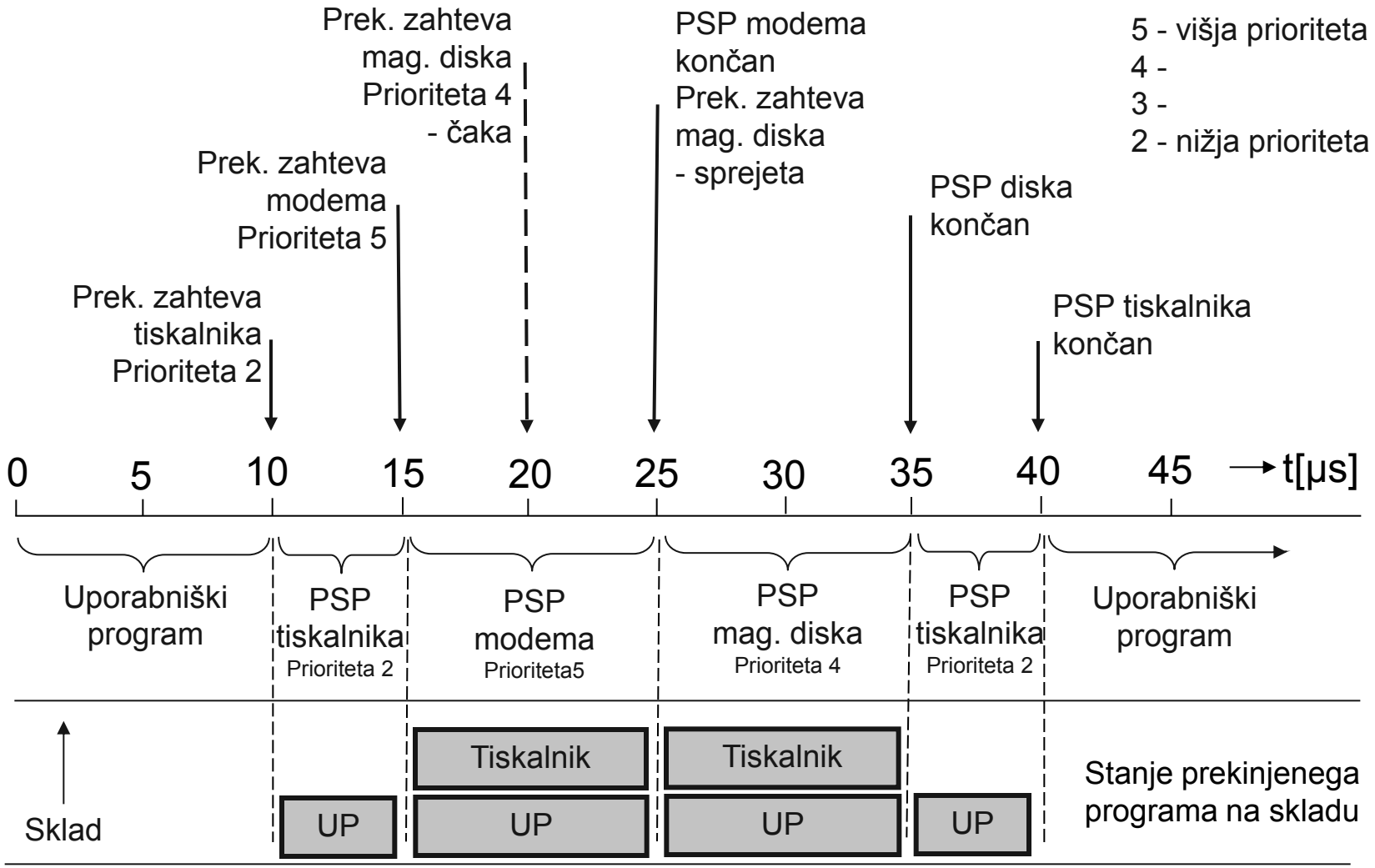
INTR – signal za prekinitveno zahtevo
INTA – signal za potrditev prekinitve



Potrjevanje prekinitvene zahteve

- Obveščanje V/I naprave, da je bila njena prekinitvena zahteva sprejeta
- Potrjevanje je potrebno, ker ni mogoče vnaprej napovedati, kako hitro bo CPE reagirala na prekinitveno zahtevo.
- Zato mora krmilnik V/I naprave vzdrževati prekinitveno zahtevo aktivno, dokler CPE ne potrdi, da je sprejela prekinitveno zahtevo.

- Potrjevanje je lahko programsko ali strojno.
  - Pri programskem potrjevanju PSP bere ali piše v neki register krmilnika V/I naprave, ki je zahtevala prekinitev.
  - Pri strojnem potrjevanju pa CPE s posebnim kontrolnim signalom obvesti napravo, da je njena prekinitvena zahteva sprejeta (npr. signal INTA).
- Pri določanju prioritete, prepoznavanju in potrjevanju je mogoče izbrati programsko ali strojno rešitev. Izbira rešitve vpliva na odzivni čas (čas od zahteve do začetka izvajanja PSP).
- Primer časovnega poteka vgnezdenih prekinitev z različnimi prioritetami:



3.6 Lokalnost pomnilniških dostopov in pomnilniška hierarhija

- Princip lokalnosti pomnilniških dostopov je eden najpomembnejših pojavov, ki ga opazimo pri delovanju von Neumannovega računalnika.

- Programi pogosto več kot enkrat uporabijo iste ukaze in operande.

- Programi bolj pogosto uporabljajo ukaze in operande, ki so v pomnilniku blizu trenutno uporabljanim.

- Izkustveno pravilo: tipičen program 90 % časa uporablja samo 10 % ukazov.
- Lokalnost je posledica načina delovanja von Neumannovega računalnika.
  - Ukazi se iz pomnilnika jemljejo eden za drugim po naraščajočih naslovih (če ni skoka).
- Lokalnost je tudi posledica načina pisanja programov.
  - Zanke v programih
  - V programih pogosto nastopajo podatkovne strukture, ki se uporabljajo po naraščajočih naslovih (polja …).

- **Prostorska lokalnost:** po pojavu pomnilniškega naslova *A(i)* je verjetno, da bodo naslednji naslovi, npr. *A(i+n),* blizu naslova *A(i).*
  - Če ni skokov, se ukazi berejo iz pomnilnika po naraščajočih naslovih.
  - Podatkovne strukture, procedure

- **Časovna lokalnost:** program ob nekem času *t* pogosto tvori naslove, ki jih je tvoril malo pred *t* in jih bo tvoril tudi nekoliko po *t.*
  - Zaradi zank v programih se isto zaporedje ukazov in s tem pomnilniških naslovov večkrat ponovi.

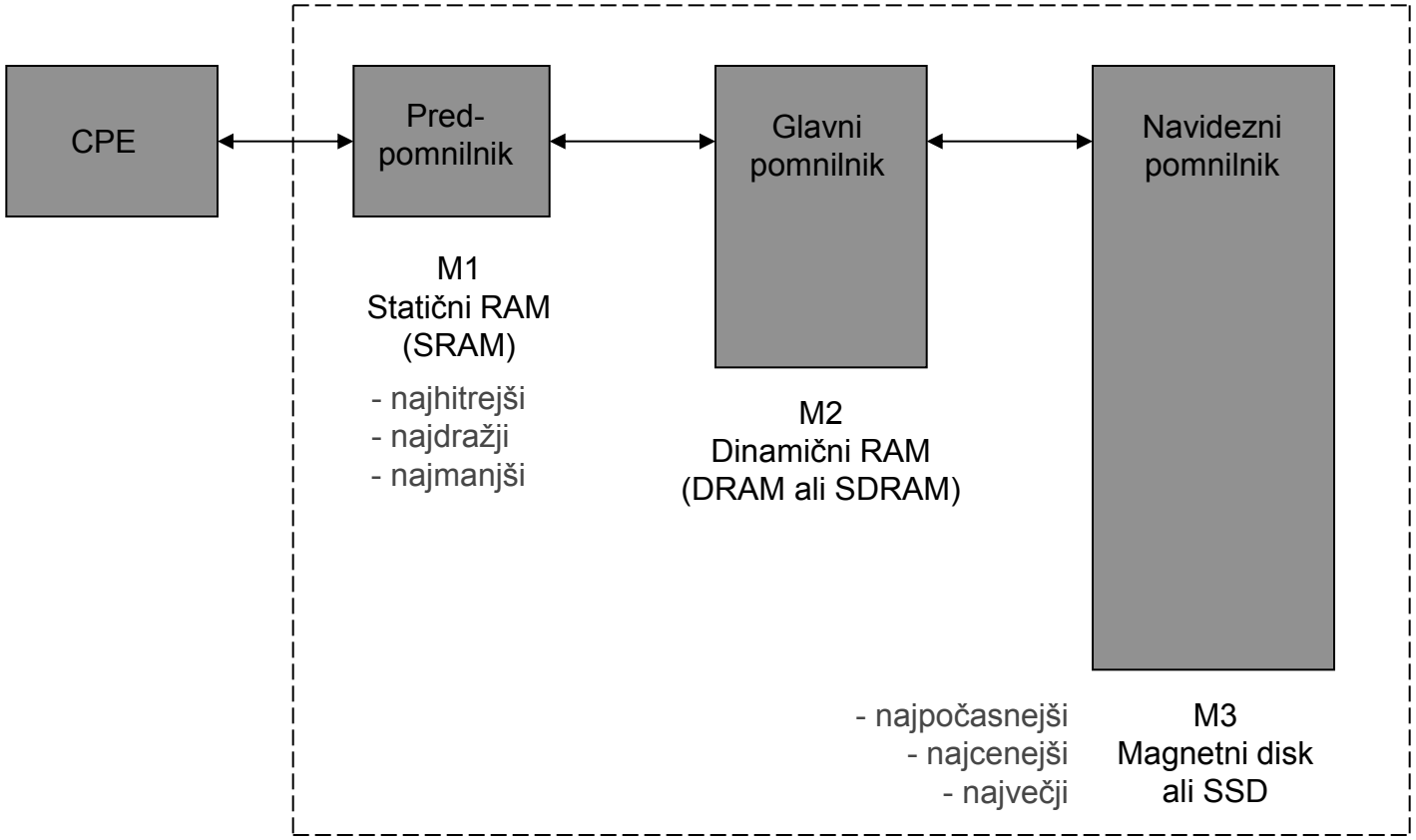
Pomnilniška hierarhija

- Lokalnost pomnilniških dostopov omogoča, da glavni pomnilnik nadomestimo s pomnilniško hierarhijo.

- Pomnilniško hierarhijo sestavlja več ločenih pomnilnikov z različnimi lastnostmi.
  - Prvi v hierarhiji – pomnilnik M1 (najbližji CPE) – je najhitrejši, najdražji in najmanjši.
  - Zadnji v hierarhiji – pomnilnik Mn (najbolj oddaljen od CPE) – je najcenejši, največji in najpočasnejši.

- Cilj pomnilniške hierarhije je, da je veliki, počasni in ceneni pomnilnik M3 videti kot hitri in dragi pomnilnik M1.

Primer trinivojske pomnilniške hierarhije



CPE vidi pomnilniško hierarhijo kot glavni pomnilnik, kot je definiran v von Neumannovem modelu.

Delovanje pomnilniške hierarhije

- Pravilo delovanja hierarhije je, da je pomnilniški prostor na nivoju i podmnožica prostora na nivoju i+1.

- Če informacije, do katere želi CPE, ni v M1, se mora prenesti iz M2 v M1. Če je ni tudi v M2, se najprej prenese iz M3 v M2 in nato iz M2 v M1.

- Prenašanje iz enega nivoja na sosednji nivo se izvaja avtomatsko, brez sodelovanja programerja.

- Za CPE je trinivojska pomnilniška hierarhija videti kot glavni pomnilnik velikosti M3 in s hitrostjo, ki je blizu hitrosti M1 (povprečna hitrost).

- Pomnilniška hierarhija bi bila brez pojava lokalnosti pomnilniških dostopov neuporabna.

Lastnosti pomnilnikov v pomnilniški hierarhiji

Tehnologija	Cena $/GB	Dostopni čas v ns	Hitrost prenosa v GB/s
SRAM	10.000 $	1 ns	25+ GB/s
DRAM	10 $	10 - 50 ns	10 GB/s
SSD	1 $	100.000 ns	0,5 GB/s
HD	0,1 $	10.000.000 ns	0,1 GB/s

SRAM – Statični RAM (bralno-pisalni pomnilnik)
DRAM – Dinamični RAM
SSD – Solid State Disk (polprevodniški disk)
HD – Hard disk (magnetni disk)

3.7 Amdahlov zakon (1967)

- G. M. Amdahl je eden od arhitektov slavne serije računalnikov IBM 370.

- Če v računalniku za faktor *N* (N-krat) pohitrimo delovanje pri vseh operacijah, razen pri *f*-temu delu od vseh operacij, potem je povečanje hitrosti celotnega računalnika *S(N)* enako:

$$S(N) = \frac{1}{f + \frac{1-f}{N}} = \frac{N}{1 + (N-1) * f}$$

S(N) = povečanje hitrosti celotnega sistema
N = faktor povečanja hitrosti (1 – f)-tega dela
f = delež operacij, ki niso pohitrene
1 – f = delež operacij, ki so N-krat pohitrene

- Primer:

- Izvajanje programov na nekem računalniku bi želeli pohitriti tako, da enojedrni procesor zamenjamo z osemjedrnim (8 paralelno delujočih CPE).

- Kolikokrat hitreje se bodo izvajali programi, če se lahko paralelno izvaja 60 % programov?

- *N = 8; 1 – f = 0,6; f = 0,4*

$$S(N) = \frac{8}{1 + (8-1) * 0{,}4} = \frac{8}{1 + 2{,}8} = 2{,}1$$

- Hitrost izvajanja programov se poveča za faktor 2,1 (2,1-krat).

- Če so se programi pred zamenjavo izvajali npr. 100 sekund, se bodo potem izvajali 47,6 sekunde (100 / 2,1 = 47,6).

Case/Amdahlovi pravili

- Nastali sta pri razvoju IBM 370. Računalnik je dobro zasnovan, če sta izpolnjeni zahtevi dveh pravil:
  - **Prvo Case/Amdahlovo pravilo:** velikost glavnega pomnilnika v bajtih mora biti najmanj enaka številu ukazov, ki jih CPE izvede v eni sekundi.
  - **Drugo Case/Amdahlovo pravilo**: zmogljivost V/I sistema v bitih na sekundo mora biti najmanj enaka številu ukazov, ki jih CPE izvede v eni sekundi.

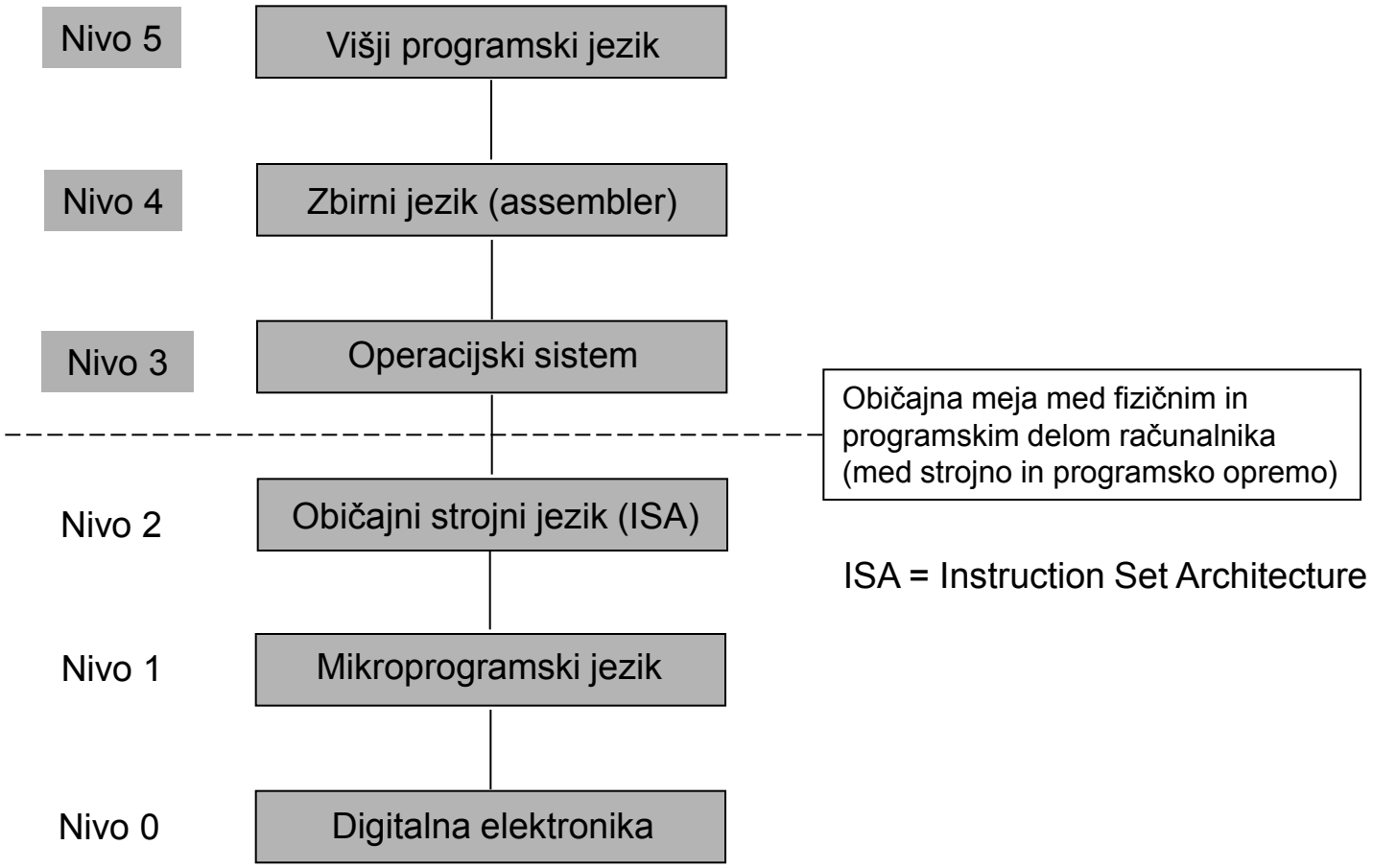
3.8 Jeziki, nivoji in navidezni računalniki

- Za veliko večino uporabnikov so podrobnosti o zgradbi in delovanju računalnikov nepomembne.
- Računalnik in njegove lastnosti vidijo predvsem skozi lastnosti programskega jezika, ki ga uporabljajo.
- Neki programski jezik se lahko realizira na zelo različnih računalnikih, to pa pomeni, da so različni računalniki za uporabnika, ki uporablja ta programski jezik, videti bolj ali manj enaki.

Računalnik kot zaporedje navideznih računalnikov

- Pri veliki večini današnjih računalnikov imamo 6 nivojev.
- Na vsakem nivoju vidimo računalnik skozi drugačen programski jezik.
- Ta programski jezik si lahko uporabnik predstavlja kot strojni jezik nekega navideznega računalnika.
- Na najnižjem nivoju (nivo 0) elektronika (logična vrata in flip-flopi) neposredno izvaja najenostavnejše ukaze.

Računalnik s šestimi nivoji



- Nivo 1 lahko zasledimo pri mnogih današnjih računalnikih. RISC računalniki nimajo nivoja 1.
  - Vsak ukaz običajnega strojnega jezika se izvrši kot zaporedje mikroukazov – računalnikom, ki tako delujejo (imajo nivo 1), rečemo, da so mikroprogramirani.
  - Pri teh računalnikih je mikroprogramski jezik dejansko pravi strojni jezik.
  - Ker v začetku računalniki tega nivoja niso imeli in je za uporabnika neviden, se pojem strojni jezik uporablja na nivoju 2.
  - Mikroprogram na nivoju 1 je napisan pri proizvajalcu in pravzaprav definira običajni strojni jezik. Uporabnik ga običajno ne more spreminjati.

- Uporabnik vidi računalnik na nivoju 2 skozi uporabo običajnih strojnih ukazov, ki tvorijo običajni strojni jezik.
  - Računalniška arhitektura je določena z zgradbo in lastnostmi računalnika, kot jih vidi programer na tem nivoju (glej RA-1/str. 21). Zato tudi ime ISA – Instruction Set Architecture.
  - Z običajnim strojnim jezikom ima programer popoln nadzor nad vsemi deli računalnika.
  - Pri prvih računalnikih višjih nivojev sploh ni bilo in je programiranje potekalo samo v običajnem strojnem jeziku.

- Nivo 3 je nivo operacijskega sistema.
  - Jezik na tem nivoju vsebuje vse ukaze nivoja 2, ki so jim dodani novi ukazi za lažje delo z računalnikom (npr. delo z V/I napravami, paralelno izvajanje programov, diagnostični ukazi).
  - Operacijski sistem je program, ki olajša delo z računalnikom in služi kot vmesnik med uporabnikom in strojno opremo računalnika.
  - Z operacijskim sistemom želimo doseči:
    - lažje delo,
    - boljši izkoristek strojnih zmogljivosti računalnika (v določenem času opraviti kar največ dela).

- Funkcije operacijskega sistema bi bilo mogoče realizirati tudi strojno na nivoju 2, vendar je programska izvedba bolj ekonomična (več operacijskih sistemov, nadgradnja . . .).
- Na tem nivoju je običajna tudi delitev uporabnikov z različno pravico uporabe ukazov.
- Nekateri ukazi nivoja 2 so običajnim uporabnikom na nivoju 3 nedostopni (dostopni samo sistemskim programerjem).
- Za večino današnjih programerjev je nivo 3 najnižji nivo, na katerem lahko delajo.

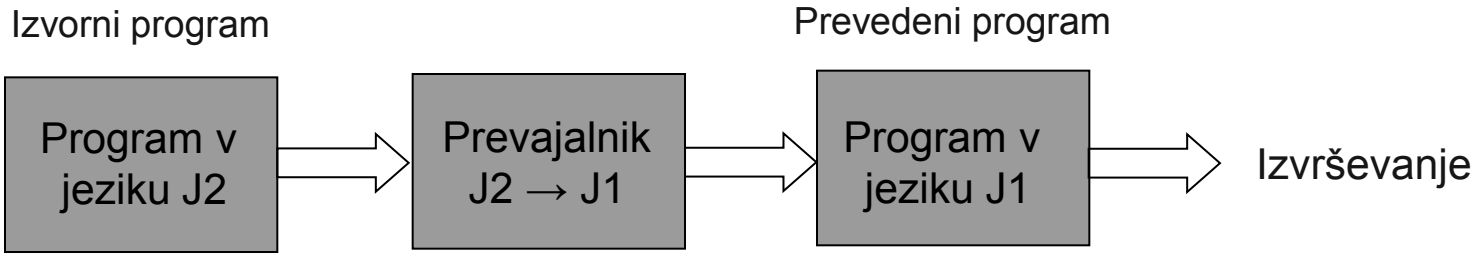
- Na nivoju 4 uporabnik vidi računalnik skozi zbirni jezik.
  - Zbirni jezik je samo simbolična, človeku bližja oblika jezika nivoja 3 (in s tem tudi nivoja 2).
  - Programe v zbirnem jeziku je treba pred izvajanjem prevesti na jezik nivoja 3 (oziroma 2).
- Nivo 5 oblikujejo višji programski jeziki, ki so namenjeni večini programerjev.
  - To so npr. C, C++, JAVA, BASIC, FORTRAN, COBOL in mnogi drugi.
  - Programe, napisane v teh jezikih, je treba prevesti na jezik nivoja 4 ali nivoja 3.

- V računalnikih lahko ugotovimo tudi višje nivoje, kot npr. program za delo s podatkovnimi bazami.
- Vsak nivo si lahko predstavljamo kot navidezni računalnik, ki ima za strojni jezik kar jezik tega nivoja.
- Vsekakor pa je treba programe, napisane v jeziku navideznega računalnika, pretvoriti v zaporedje ukazov strojnega jezika.
- Uporabniki se tega pretvarjanja pogosto ne zavedajo, proizvajalci računalnikov pa morajo poskrbeti za prehajanje iz enega jezika v drugega.

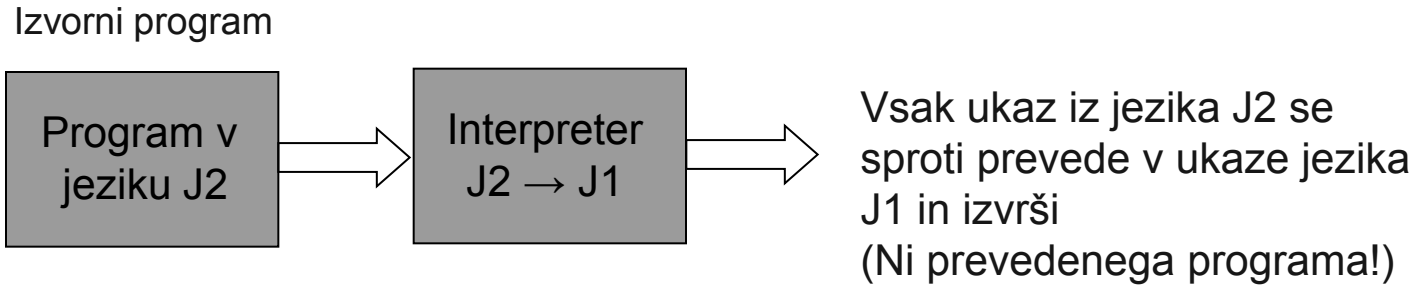
- Mehanizem prehajanja iz enega jezika v drugega je lahko realiziran na dva načina:
  - s prevajanjem,
  - z interpretiranjem.
- Po letu 1990 pa se je razširila še vmesna rešitev:
  - delno prevajanje.
- Glavna razlika med prevajanjem in interpretiranjem je, da pri interpretiranju ne obstaja prevedeni (ciljni) program.

Prehajanje iz jezika J2 v jezik J1

Prevajanje



Interpretiranje



- Prevedeni programi delujejo samo na računalniku s strojnim jezikom, v katerega so bili prevedeni.
- Pred prenosom na drugačen računalnik (z drugačnim strojnim jezikom L1) je treba izvorni program znova prevesti.

- Z vključevanjem velikega števila različnih računalnikov v omrežja je postala prenosljivost programov, ki jo omogoča interpretiranje, zelo pomembna.

- Delno prevajanje je neka vmesna rešitev med interpretiranjem in prevajanjem, ki omogoča hitrejše interpretiranje.

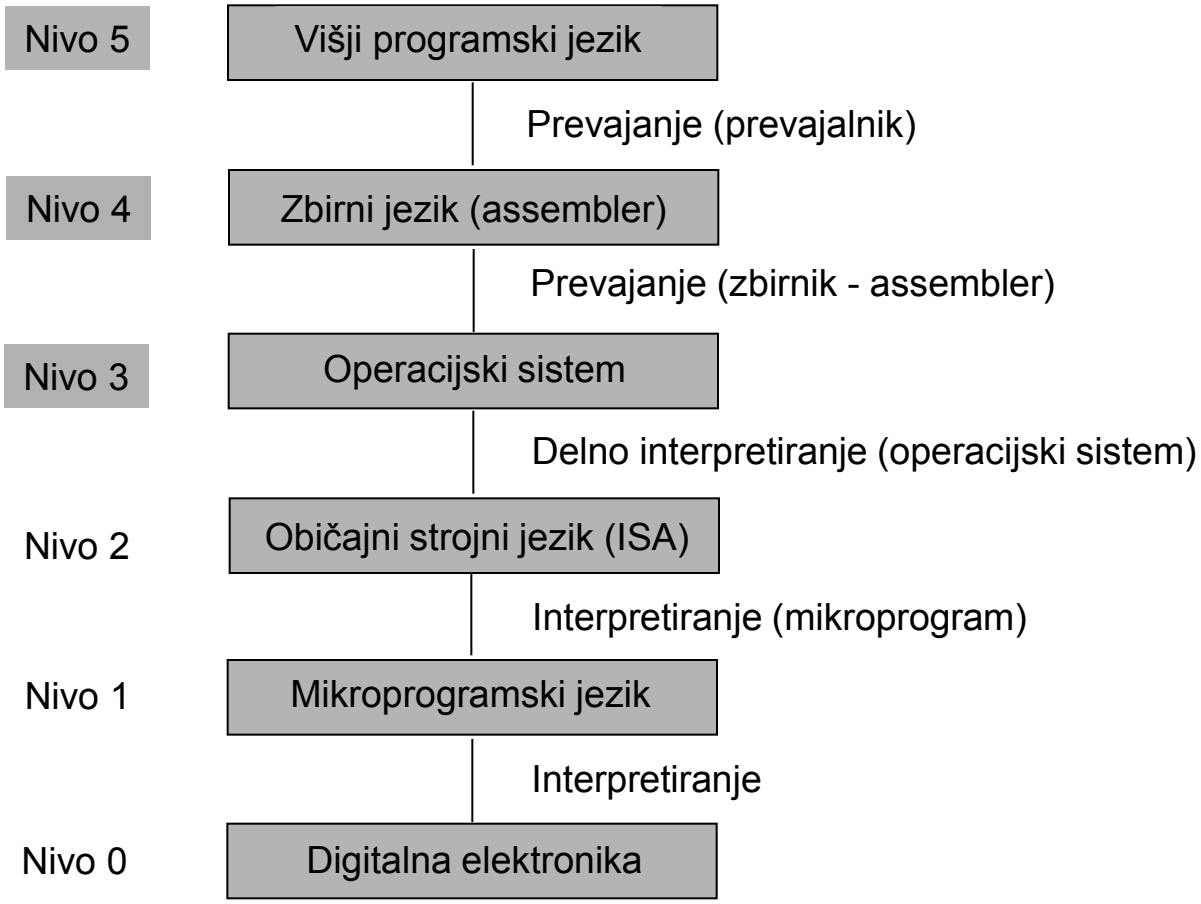
- Delno prevajanje: Izvorni program v jeziku J2 se prevede v program v vmesnem jeziku J1, program v J1 pa se interpretira.

- Delno prevajanje v vmesni jezik J1 omogoča hitrejše interpretiranje, ki pa je vseeno tipično 10-krat počasnejše kot izvajanje v celoti prevedenega programa.

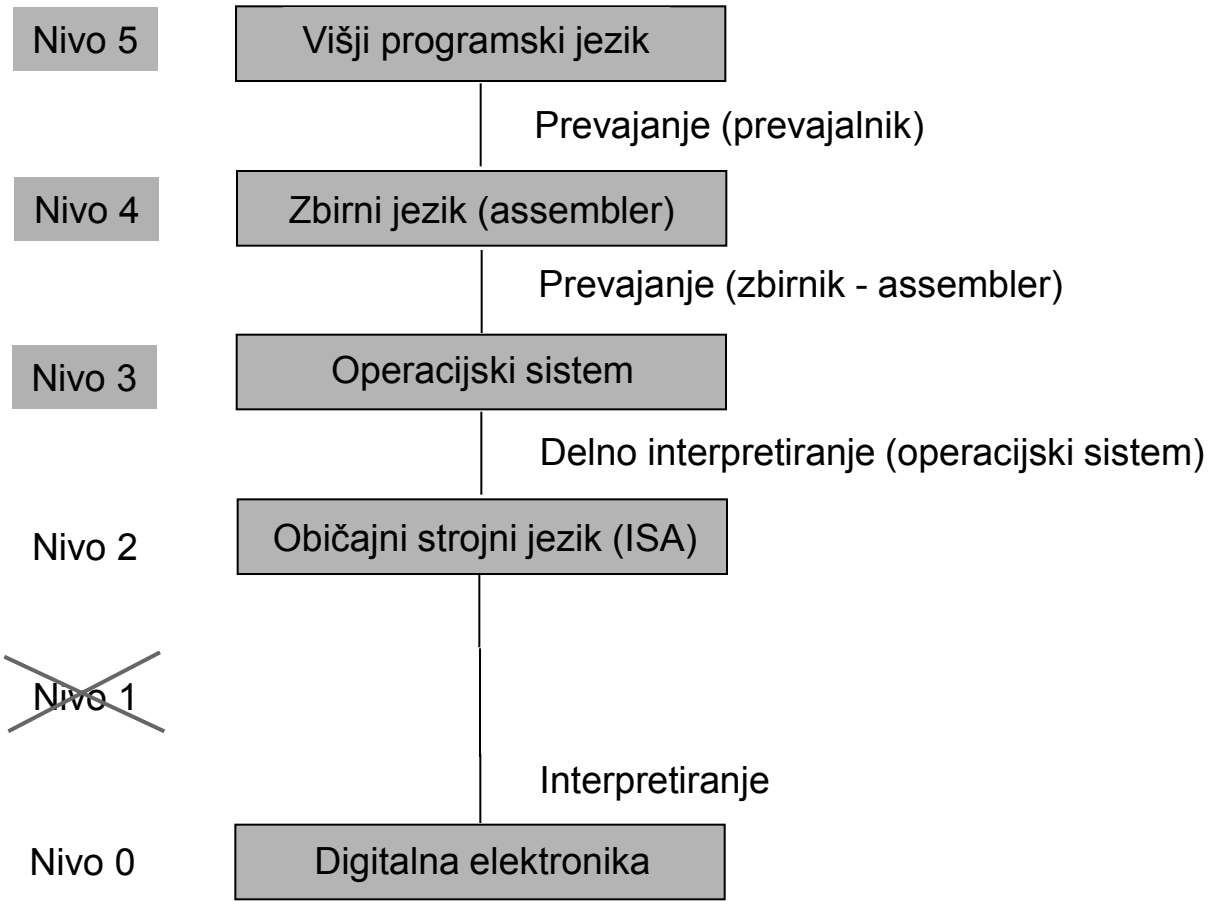
- Tako je omogočena prenosljivost programov pri bistveno manjši izgubi hitrosti, kot če bi uporabili samo interpretiranje.

- Primer navideznega računalnika: JVM (Java Virtual Machine)
  - Virtual Machine – navidezni stroj (navidezni računalnik) je programska izvedba stroja (računalnika), ki deluje (izvaja programe) enako kot realen stroj (računalnik).
  - Javanski programi se izvajajo tako, da se najprej prevedejo (delno prevajanje) v neki vmesni jezik (Java byte code), ki se interpretira s programom JVM.

Računalnik s šestimi nivoji (mikroprogramiran)



Računalnik s petimi nivoji (RISC računalnik)



Strojna in programska oprema računalnika

- Meja med strojnim in programskim delom računalnika ni trdna - lahko jo premikamo.
- Vsakega od nivojev lahko realiziramo tako strojno kot tudi programsko.
- Nivo 2 je npr. lahko realiziran s programom, ki teče na drugem računalniku.

Strojna in programska oprema sta logično ekvivalentni.

- Vsaka operacija, ki jo izvede programska oprema, se lahko realizira tudi direktno strojno (hardversko).
- Prav tako pa vsak strojni ukaz, ki ga izvaja hardver, lahko simuliramo s programom.
- Razvoj večnivojskih strojev
  - Iznajdba mikroprogramiranja (1951)
  - Iznajdba operacijskega sistema (okrog 1960)
  - Selitev funkcionalnosti v mikroprogram (okrog 1970)
  - Opuščanje mikroprogramiranja (po 1984)
  - Kombinacija: kompleksni ukazi običajnega strojnega nivoja so realizirani mikroprogramsko, enostavnejši ukazi so realizirani strojno.

