UI prijenos

Povezivanje računala s okolinom

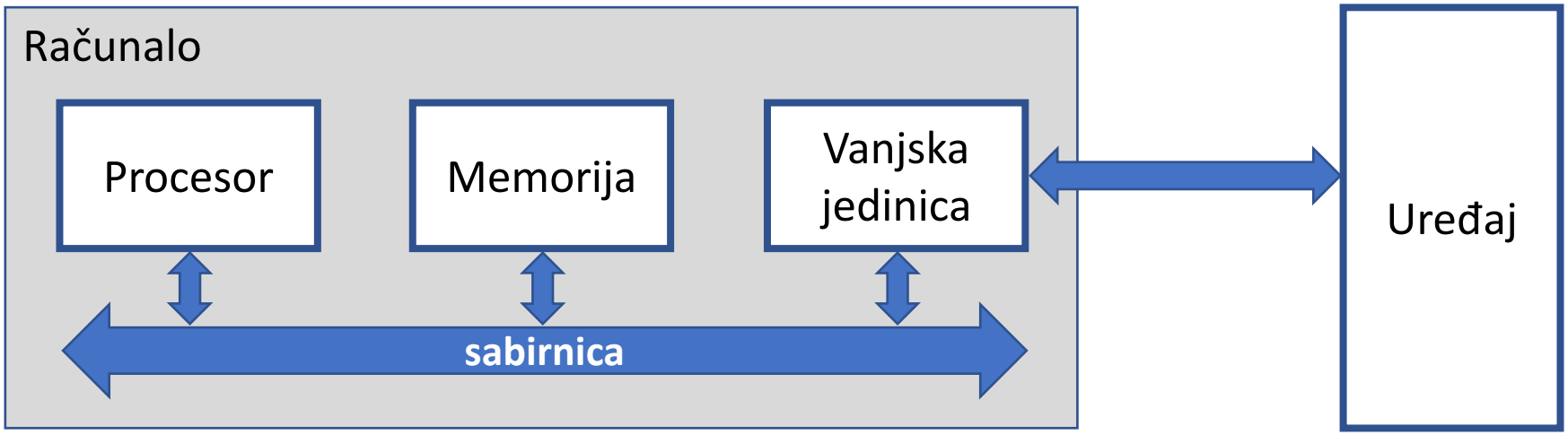
Ulaz-izlaz i vanjski uređaji

- Ulaz-izlaz ili UI (Input/Output, IO) je dio računala preko kojega računalo komunicira s korisnikom ili se povezuje s okolinom
- Periferni/vanjski uređaji (*peripherals*, *peripheral devices*, *external IO units*...) su uređaji za komunikaciju koji se spajaju na ulaz-izlaz, a nisu dio samog računala* (iako fizički mogu biti integrirani u računalo)
  - Vanjski uređaji za komunikaciju s korisnikom
    - Ulazni: miš, tipkovnica (integrirana u laptopu), mikrofon, *touchpad*, skener
    - Izlazni: zaslon (integriran u laptopu), pisač, zvučnici, slušalice, projektor
    - Dvosmjerni: zaslon osjetljiv na dodir, *gamepad*
  - Vanjski uređaji za komunikaciju s okolinom
    - Ulazni: senzori fizikalnih veličina, AD-pretvornici, kamera, bar-kod čitači
    - Izlazni: DA pretvornici, releji, pulsni motori, pojačala
    - Dvosmjerni: mreža, modemi, memorijske kartice i stickovi, eksterni diskovi, CD/DVD-ROM, pametni telefoni i satovi koji su i sami neovisna računala

* Granica je ovdje pomalo nedefinirana

Ulaz-izlaz i vanjski uređaji

- Uređaji se nikada ne spajaju izravno na sabirnicu procesora već preko ulaza i izlaza, tj. preko posebnih međusklopova koji služe kao posrednici ili sučelje prema uređaju
- Ove međusklopove zovemo vanjskim jedinicama ili UI jedinicama
  (*IO unit*, *IO interface*, *IO module*...)
- Zadaća vanjske jedinice je oslobađanje procesora od učestale komunikacije s uređajem i prilagodba načina komunikacije uređaja komunikaciji putem sabirnice računala



Vanjske jedinice

- Kakve sve mogu biti vanjske jedinice?

- Podjela prema namjeni:
  - za prijenos podataka
  - za mjerenje vremena
  - za brojanje impulsa
  - za generiranje impulsa
  - za AD i DA pretvorbu
  - specijalne namjene, itd.
- Podjela prema smjeru prijenosa:
  - ulazne
  - izlazne
  - dvosmjerne

Komunikacija procesora s vanjskim jedinicama

Adresiranje vanjskih jedinica

Načini UI prijenosa

Adresiranje vanjskih jedinica

- Načelno, kod komunikacije procesora s vanjskim jedinicama postoje dvije operacije kao i kod pristupanja memoriji:
  - čitanje
  - pisanje

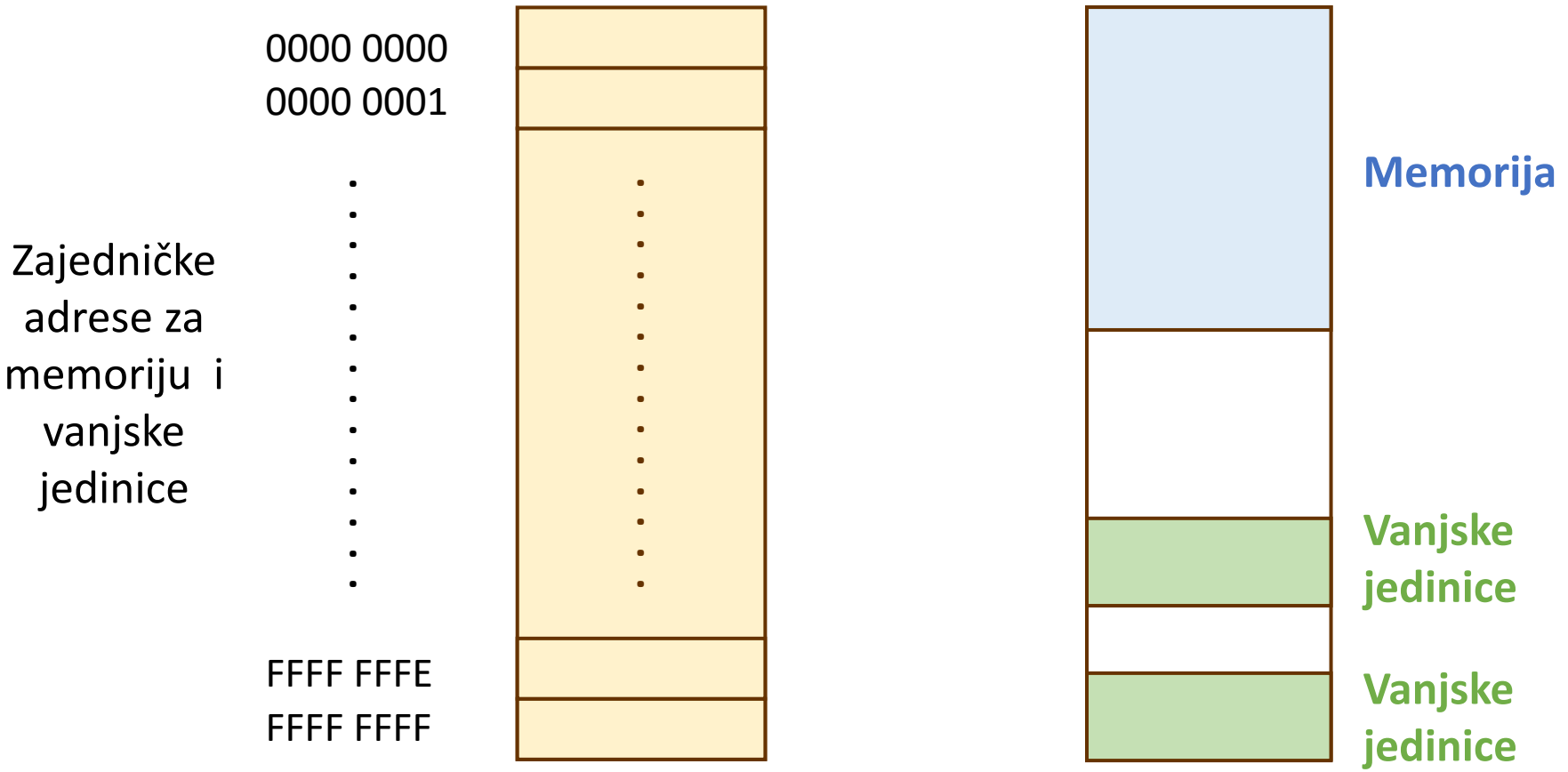
- Na sabirnicu je spojen veći broj vanjskih jedinica i procesor mora odabrati onu jedinicu s kojom želi komunicirati - to se radi pomoću adresne sabirnice

- Dva osnovna načina adresiranja vanjskih jedinica su:
  - memorijsko adresiranje ulaza/izlaza (*memory mapped IO*)
  - izdvojeno adresiranje ulaza/izlaza (*isolated IO*)

Memorijsko adresiranje UI

LOGIČKI ADRESNI PROSTOR

FIZIČKI ADRESNI PROSTOR



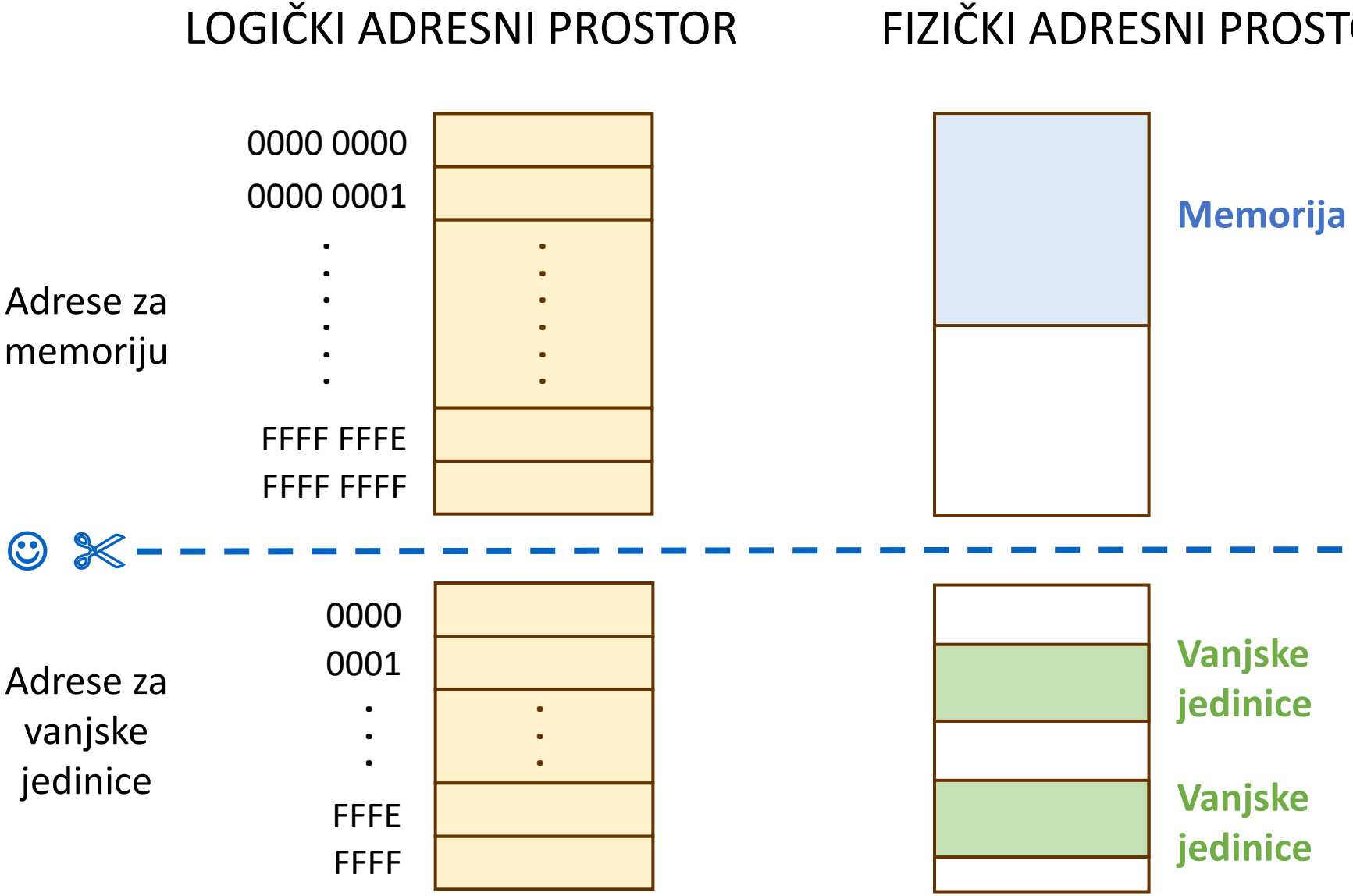
Memorijsko adresiranje UI

- Memorijsko adresiranje UI ima sljedeće značajke:

- procesor pristupa memorijskim lokacijama i vanjskim jedinicama korištenjem istih naredaba
  - npr. LOAD i STORE - pogodno za RISC procesore

- sabirnički protokoli jednaki su za pristup memoriji kao i za pristup vanjskim jedinicama

Izdvojeno adresiranje UI



Izdvojeno adresiranje UI

- Izdvojeno adresiranje UI ima sljedeće značajke:

- procesor ima različite naredbe za pristup memoriji od onih za pristup vanjskim jedinicama
  - npr. LOAD i STORE za pristup memoriji te IN i OUT za pristup vanjskim jedinicama - pogodnije za CISC procesore
- procesor ima posebne priključke kojima određuje je li neka adresa (npr. 10), koja se nalazi na adresnoj sabirnici, namijenjena memoriji ili vanjskoj jedinici
  - npr. posebna linija MEM/IO* ili dvije linije MEMREQ i IOREQ
- za pristup memoriji koriste se različiti sabirnički protokoli od onih za pristup vanjskim jedinicama

Usporedba

- Prednosti i nedostatci memorijskog i izdvojenog adresiranja:

Memorijsko adresiranje UI

- jedan sabirnički protokol
- bez dodatnih priključaka
- bez dodatnih naredaba
- manji adresni prostor
- „miješanje adresa”

Izdvojeno UI adresiranje

- više sabirničkih protokola
- dodatni priključci
- dodatne naredbe
- puni adresni prostor
- bez „miješanja adresa”

Načini UI prijenosa

- Vanjske jedinice i memorija su izvori i odredišta prijenosa
  - Procesor može privremeno čuvati podatke ili ih transformirati, ali on sam po sebi nije ni izvor ni odredište podataka
- Prijenos se može obavljati na više načina:
  - **programski (ili programirani) prijenos**
    - bezuvjetni
    - uvjetni
    - prekidni*
  - **sklopovski prijenos**
    - DMA jedinice

* U većini literature se tretira kao zasebna vrsta prijenosa, a ne kao dio programskog prijenosa

Načini UI prijenosa

- Vanjske jedinice obično su **sporije** od procesora
- Prije prijenosa, procesor i vanjska jedinica obično se trebaju **sinkronizirati** da bi se prijenos podataka uspješno izveo
  - Ovisno o načinu prijenosa, sinkronizacija se može obaviti na razne načine (programski, sklopovski, a u određenim slučajevima se može i ispustiti)
  - Vanjska jedinica se također mora sinkronizirati s uređajem, a to vanjska jedinica obično radi automatski
  - Kod nekih jednostavnijih vanjskih jedinica programer se mora sam brinuti i za sinkronizaciju s uređajem

Programski prijenos

- Glavne karakteristike programskog prijenosa su:
  - Prijenos se obavlja pod upravljanjem programa, tj. prijenos obavlja procesor (upravljan programom)
  - Svi podatci koji se prenose ”prolaze kroz procesor”
  - Programski prijenos je sporiji od sklopovskog prijenosa
  - Procesor dio vremena troši na prijenos podataka što usporava izvođenje ostalih poslova

Sklopovski prijenos

- Glavne karakteristike **sklopovskog prijenosa** su:
  - prijenos se obavlja pod upravljanjem specijaliziranog sklopovlja, tj. **prijenos obavlja specijalna jedinica** (DMA kontroler), a procesor ne sudjeluje u prijenosu
  - podatci koji se prenose prolaze kroz specijalnu jedinicu, a ovisno o njenoj građi i organizaciji moguće je čak i da ona upravlja izravnim prijenosom između vanjske jedinice i memorije
  - prijenos se općenito odvija **velikim brzinama**
  - procesor ne troši vrijeme na prijenos podataka, ali **izvođenje programa je ipak usporeno** za vrijeme obavljanja prijenosa
  - kasnije ćemo vidjeti jednu vrstu sklopovskog prijenosa:
    - **izravni pristup memoriji (DMA)**

Sinkronizacija prijenosa podataka između vanjskih jedinica i periferije

Sinkronizacija prijenosa podataka

- Sinkronizacija je uvijek potrebna kod prijenosa podataka, bez obzira radi li se od prijenosu unutar procesora, unutar računala ili između računala i okoline
- Sinkroni prijenos podataka
  - Koristi se zajednički signal *clock* koji se primjenjuje za sve prijenose podataka
  - Obično se primjenjuje interno u računalu
- Asinkroni prijenos podataka
  - Svaka komponenta ima vlastiti clock i brzinu rada
  - Sinkronizacija se ostvaruje zasebnim sinkronizacijskim signalima/linijama i može koristiti jednostavniju „jednolinijsku” dojavu ili „rukovanje” pomoću dvije linije
  - Često se koristi između računala i okoline

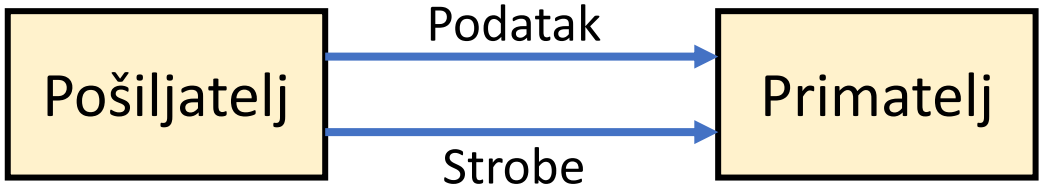
Jednolinijska sinkronizacija

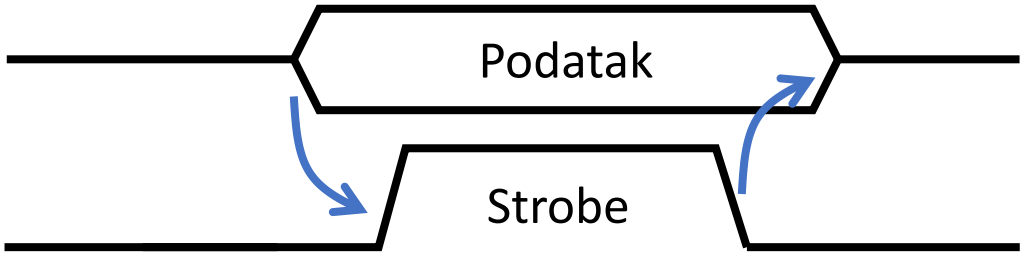
- Jednostavnija verzija asinkronog prijenosa koristi jednu liniju za sinkronizaciju uobičajeno nazvanu *Strobe*
- *Strobe* može aktivirati ili pošiljateljska (*source unit*) ili primateljska (*destination unit*) komponenta
- Pretpostavka je da „pasivna” komponenta* u trenutku aktiviranja signala *Strobe* mora biti spremna za prijenos podataka
- „Pasivna” komponenta mora obaviti traženi prijenos (primiti ili pročitati podatak) što se događa na jedan od bridova signala *Strobe*
- Nedostatak sinkronizacije pomoću signala *Strobe* je što onaj koji postavlja signal *Strobe* ne zna je li suprotna „pasivna” strana zaista spremna, tj. je li zaista primila ili poslala podatak

* Pasivna u smislu da je to ona komponenta koja **ne** aktivira *Strobe*

Strobe aktiviran od pošiljatelja

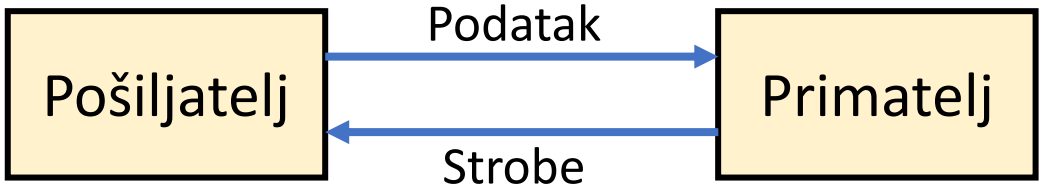
1. Pošiljatelj postavlja podatak
2. Nakon toga pošiljatelj aktivira *Strobe* da bi dojavio da je podatak valjan
3. Primatelj detektira *Strobe* i u periodu valjanosti signala Strobe mora preuzeti podatak (npr odmah ili npr na padajući brid od *Strobe*)
4. Pošiljatelj nakon nekog vremena uklanja *Strobe*, a zatim i podatak (odnosno nije određeno što se s podatkom događa)
5. Nakon toga se komunikacija vraća u početno stanje

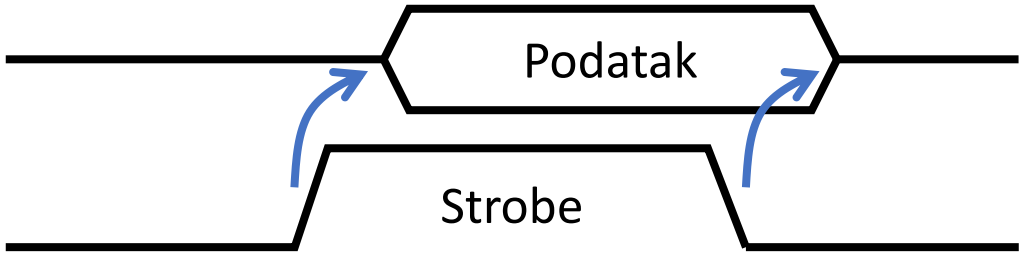




Strobe aktiviran od primatelja

1. Primatelj aktivira Strobe da bi dojavio spremnost za primitak podatka
2. Nakon toga pošiljatelj mora u određenom roku postaviti podatak
3. Primatelj će pročitati podatak te nakon čitanja deaktivirati Strobe
4. Pošiljatelj nakon deaktivacije Strobe-a može ukloniti podatak (ili ga ostaviti do sljedeće promjene)
5. Nakon toga se komunikacija vraća u početno stanje





Dvolinijska sinkronizacija rukovanjem

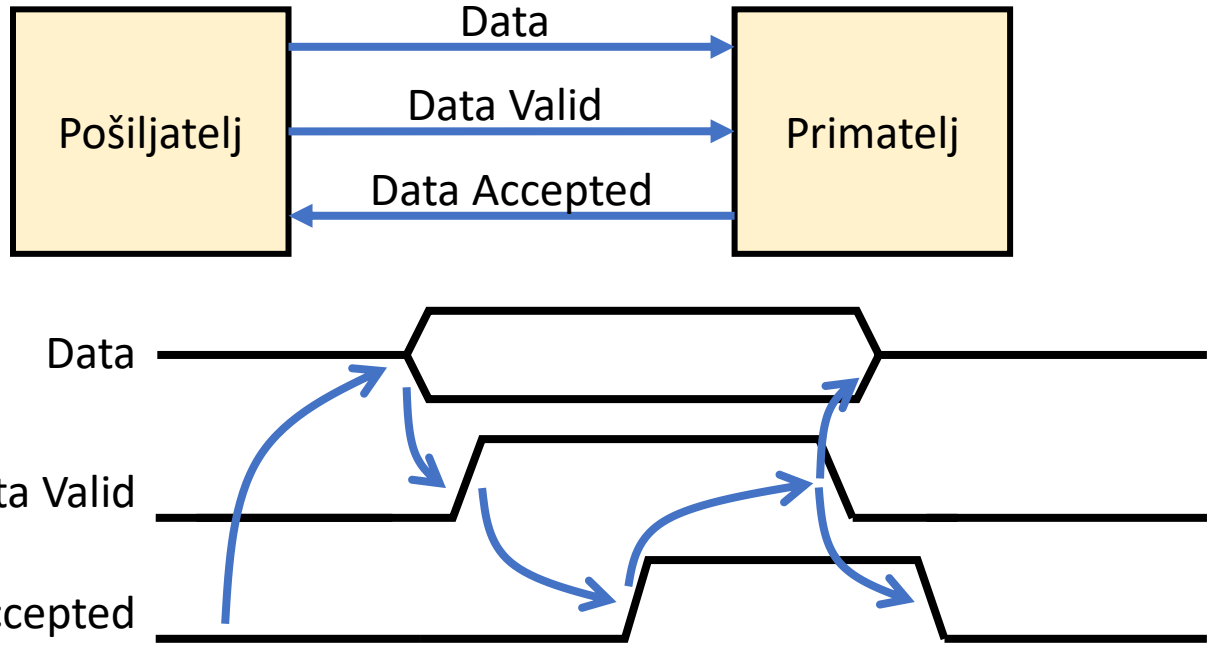
- Problem kod jednolinijske sinkronizacije je što inicijator signala *Strobe* ne zna je li suprotna strana spremna za komunikaciju
- Rješenje: protokol **rukovanja** (*handshaking*)
- Za rukovanje postoje dva signala* tako da uvijek postoji povratna informacija
- Strane koje komuniciraju prelaze na sljedeći korak komunikacije tek kad dobiju potvrdu da je suprotna strana spremna
- Ponovno su moguće varijante gdje pošiljatelj inicira prijenos ili gdje primatelj inicira prijenos

---

* Često se zovu *Ready* i *Strobe*

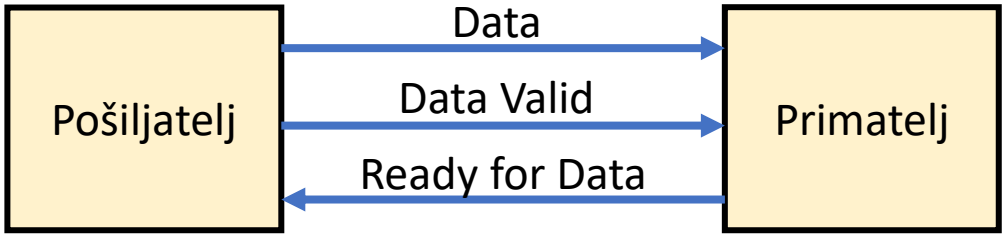
Rukovanje koje inicira pošiljatelj

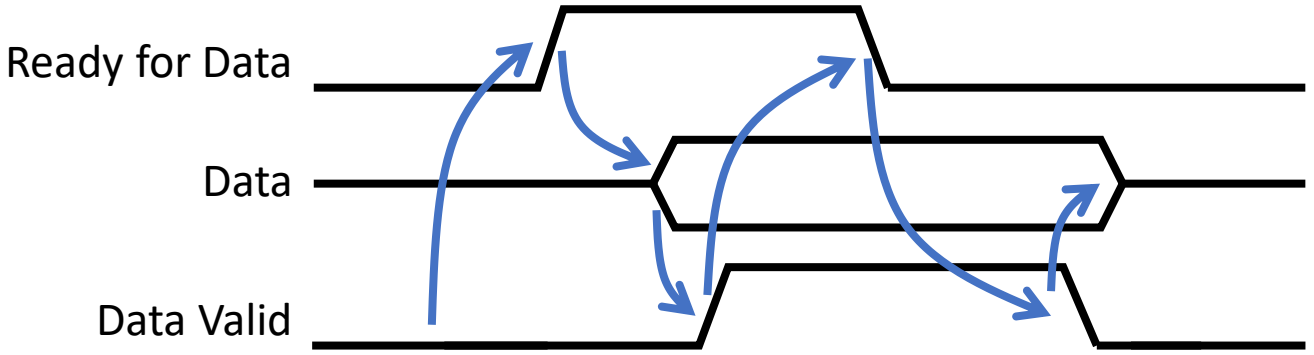
1. Kada je Data Accepted neaktivan, pošiljatelj smije postaviti podatak i signalizirati to aktiviranjem Data Valid
2. Primatelj detektira da je podatak valjan. Kada primatelj bude spreman, preuzima podatak te obavještava o tome pošiljatelja aktiviranjem Data Accepted
3. Pošiljatelj na aktiviranje Data Accepted deaktivira Data Valid i uklanja podatak (ili ga ostavlja do sljedeće promjene)
4. Primatelj na deaktivaciju Data Valid deaktivira Data Acepted



Rukovanje koje inicira primatelj

1. Kada je data Valid neaktivan, primatelj smije dojaviti da je spreman za prihvat podataka tako da aktivira Ready for Data
2. Na ovu dojavu pošiljatelj, kada bude spreman, postavlja podatak i dojavljuje njegovu valjanost aktiviranjem Data Valid
3. Primatelj detektira Data Valid i preuzima podatak te o tome obavještava pošiljatelja deaktiviranjem Ready for Data
4. Pošiljatelj, nakon što detektira da je Read for Data opet neaktivan, deaktivira Data Valid i uklanja podatak (ili ga ostavlja do sljedeće promjene)



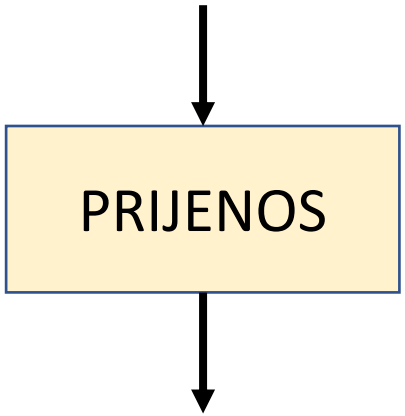


Programirani UI prijenos

Bezuvjetni
Uvjetni
Prekidni

Bezuvjetni prijenos

- Bezuvjetni prijenos je najjednostavnija vrsta prijenosa
- Glavna značajka je da se prije prijenosa ne provjerava je li vanjska jedinica spremna za prijenos podatka
  => nema sinkronizacije
- Dijagram toka je trivijalan i sastoji se samo od prijenosa
  - prijenos je jedna naredba (eventualno nekoliko njih) za čitanje ili pisanje u vanjsku jedinicu



Bezuvjetni prijenos

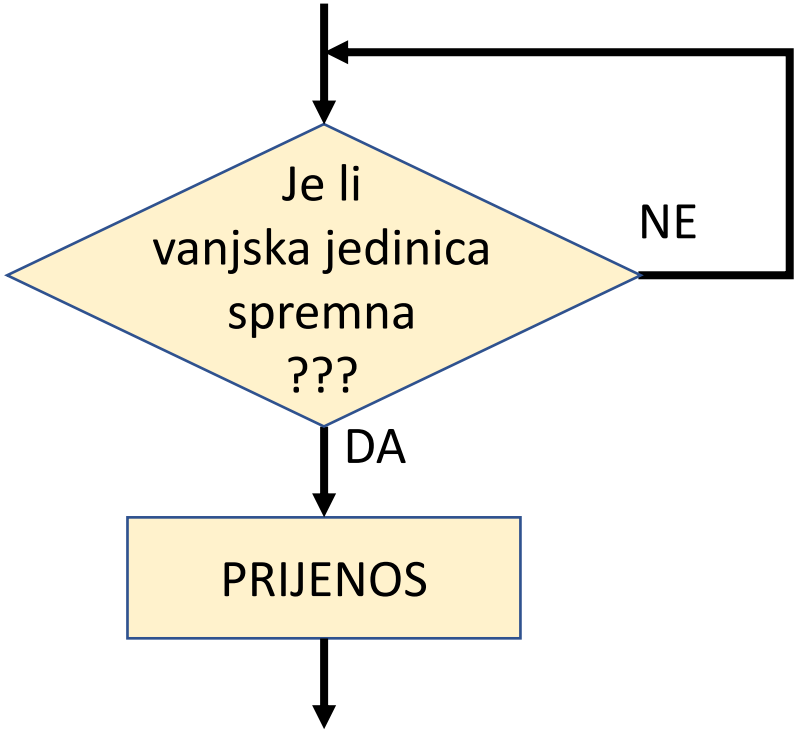
- Vanjska jedinica je sklopovski **najjednostavnija**
- **Najbrži i najjednostavniji** prijenos među programiranim prijenosima
- Nedostatak je mogućnost da **vanjska jedinica nije spremna za prijenos**
  - slanje na nespremnu jedinicu može „pregaziti” prethodno poslani podatak koji će biti izgubljen ili se pak novi poslani podatak može zanemariti/izgubiti
  - primanje s nespremne jedinice može ponovno pročitati prethodni podatak ili će se pročitati podatak koji nije ispravan

Bezuvjetni prijenos

- Kada možemo koristiti bezuvjetni prijenos?
- Općenito:
  - kada nam nije bitna sinkronizacija ili ona ne postoji
  - kada znamo da nećemo pristupati vanjskoj jedinici brže nego što ona može raditi
- Primjeri:
  - želimo s nekog jednostavnog senzora očitati trenutno stanje neke fizikalne veličine
  - želimo očitati stanje tipke (pritisnuta ili ne)
  - želimo preko nekog releja uključiti ili isključiti neki uređaj
  - želimo svake sekunde osvježiti prikaz brzine na zaslonu (npr. na 7-segmentnom display-u)

Uvjetni prijenos

- **Uvjetni prijenos** rješava probleme gubitka podataka ili rada sa starim podatcima koji su prisutni kod bezuvjetnog prijenosa
- Glavna značajka je da se prije prijenosa uvijek provjerava je li vanjska jedinica spremna za prijenos podatka
  => postoji sinkronizacija



Uvjetni prijenos

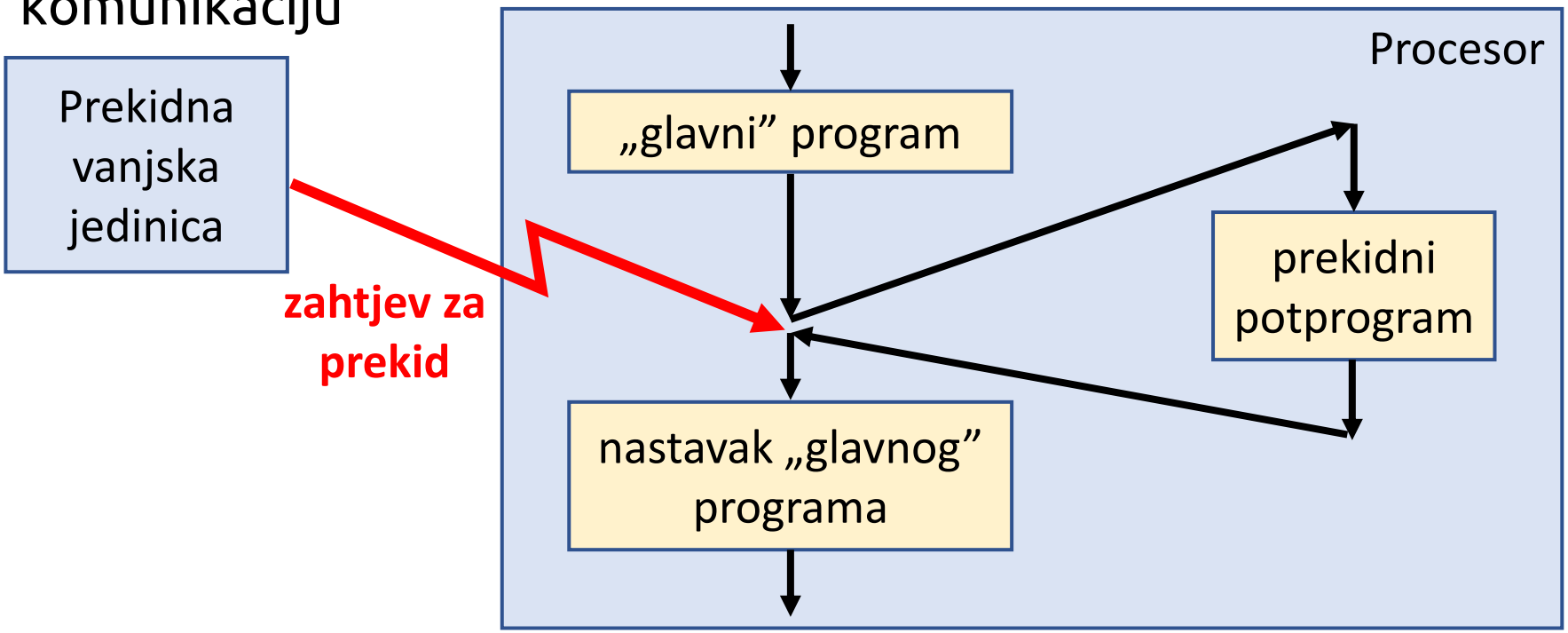
- Glavna prednost je da nema gubitaka/uvišestručenja podataka
- Glavni nedostatak je dugotrajno čekanje na spremnost uvjetne vanjske jedinice (budući da je procesor puno brži, umjesto da radi koristan posao mora vrtiti petlju za ispitivanje spremnosti)
- Uvjetna vanjska jedinica je sklopovski složenija od bezuvjetne
  - Ima stanje spremnosti koje se pamti u bistabilu stanja (*status bistabilu*)
  - Mora imati sinkronizaciju s vanjskim uređajem (bilo sklopovski izvedenu, bilo mogućnost da se izvede programski)

Uvjetni prijenos

- Kada možemo koristiti uvjetni prijenos?
- Općenito:
  - Kada nam je bitna sinkronizacija (želimo prenijeti svaki podatak točno jednom)
  - Kada ne znamo u kojem trenutku ili kojom brzinom ćemo pristupati vanjskoj jedinici pa se može dogoditi da nije spremna
  - Kada procesor nema drugih zadaća osim posluživanja vanjskih jedinica pa nije problem da gubi vrijeme na čekanje
- Primjeri:
  - želimo slati znakove na pisač (svi moraju biti ispisani)
  - želimo primati niz podataka koji dolaze s uređaja (npr. tipkovnica ili mreža) i spremati ih u memoriju
  - želimo učitati stanje senzora i želimo biti sigurni da smo pročitali valjano stanje

Prekidni prijenos

- Kod programiranog UI prijenosa određeni dio programa obavlja prijenos i svu komunikaciju s vanjskom jedinicom
- Kod bezuvjetnog i uvjetnog prijenosa procesor će pod upravljanjem programa inicirati početak prijenosa
- Bitna razlika kod prekidnog prijenosa: **početak prijenosa inicira sama vanjska jedinica** i to onda kada je spremna za komunikaciju



Prekidni prijenos

- Prekidna vanjska jedinica je sklopovski **najsloženija** (kao uvjetna + dodatna logika za prekide)
- **Nema gubitka podataka** (bolje od bezuvjetnog, isto kao uvjetni)
- **Relativno brz i efikasan prijenos** **VAŽNO!!!**
  - nema čekanja kao kod uvjetnog prijenosa
  - obrada prekida traje neko vrijeme pa je bezuvjetni prijenos ipak brži
- **„Najbolji"** prijenos među programiranim prijenosima jer predstavlja kompromis brzine i sigurnosti da se podatci ne gube

Prekidni prijenos

- Kada možemo koristiti prekidni prijenos?
- Općenito:
  - Onda kada nam je bitna sinkronizacija
    → želimo prenijeti svaki podatak točno jednom
  - Onda kada procesor mora obavljati različite druge zadaće, a ne samo posluživati vanjsku jedinicu

- Primjeri:
  - isti primjeri kao za uvjetni prijenos...
  - ... ali ako procesor nema vremena za čekanje