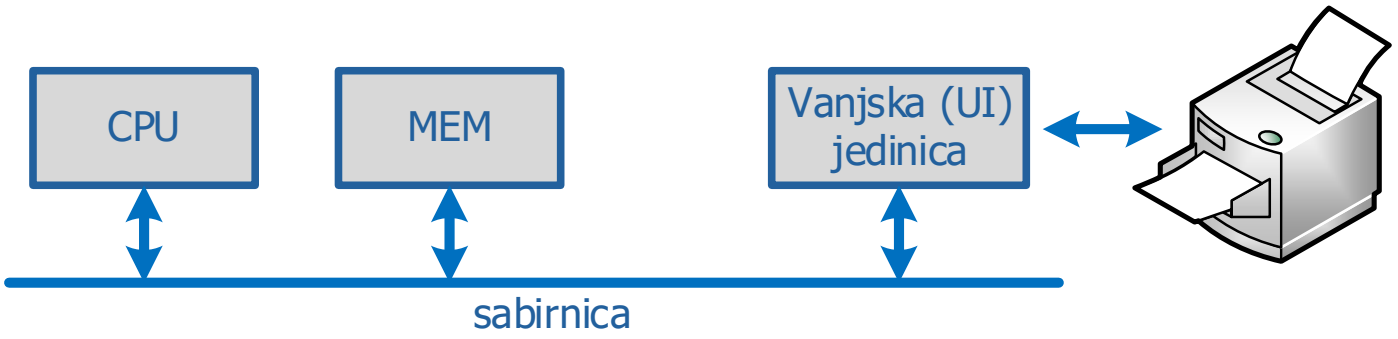
Povezivanje računala s okolinom

UI uređaji

- Uređaji se nikada ne spajaju izravno na sabirnicu procesora (razlozi !) već preko "ulaza i izlaza", tj. preko posebnih međusklopova namijenjenih upravo toj svrsi (oni služe kao posrednici)

- Ovakve međusklopove zovemo ulazno-izlaznim (UI) jedinicama (engl. IO unit) ili vanjskim jedinicama (skraćeno VJ)
- Zadaća UI jedinice je oslobađanje procesora od učestale komunikacije s uređajem i prilagodba načina komunikacije uređaja komunikaciji putem sabirnice računala



Povezivanje računala s okolinom

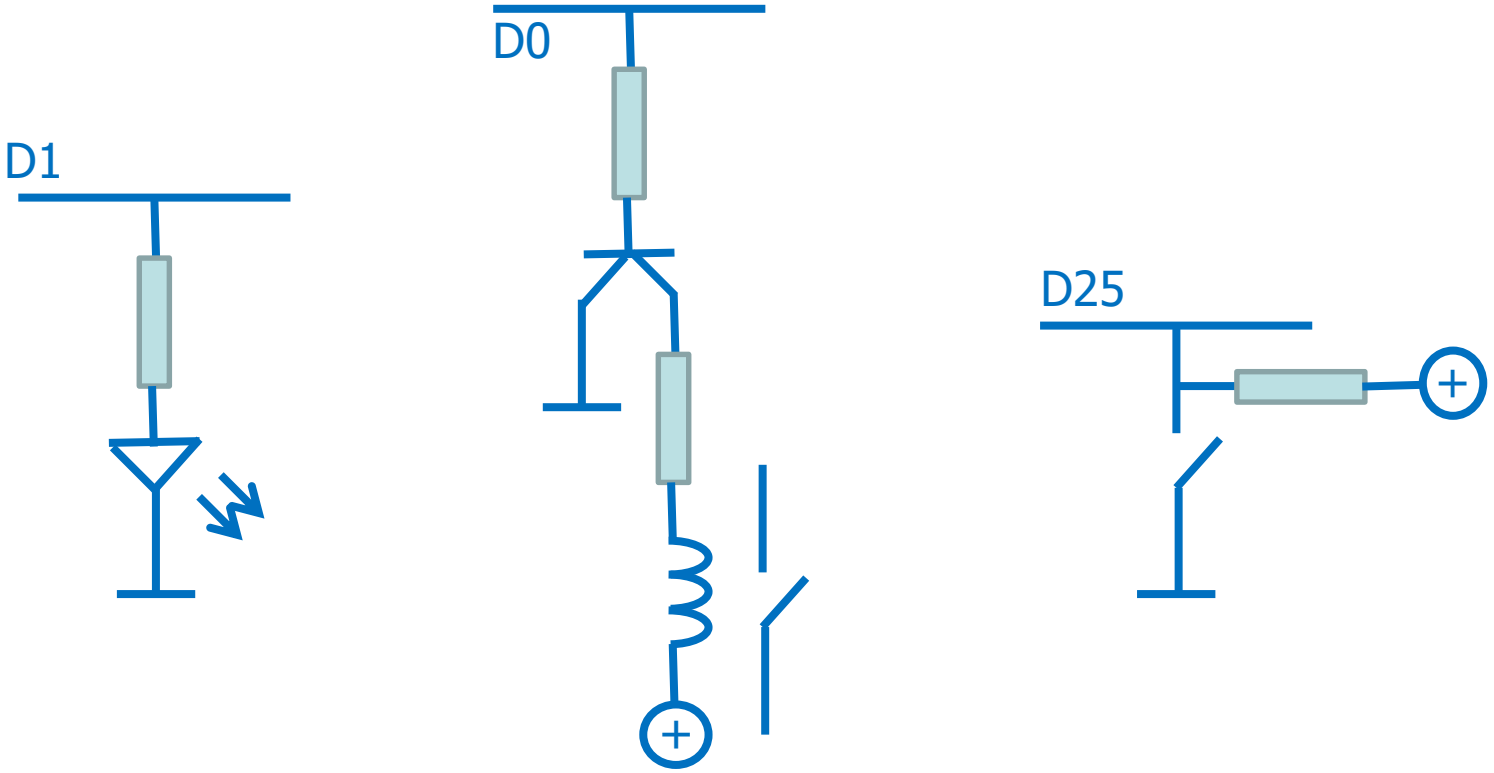
- Kakve sve mogu biti UI jedinice?
- Možemo ih podijeliti prema smjeru prijenosa:
  - ulazne
  - izlazne
  - dvosmjerne
- Možemo ih podijeliti prema namjeni:
  - za prijenos podataka
  - za brojanje impulsa
  - za mjerenje vremena
  - za generiranje impulsa
  - za AD i DA pretvorbu
  - specijalne namjene, itd.

Povezivanje računala s okolinom

- Što sve može biti uređaj?
  - Senzor preko kojeg primamo podatke o vanjskom svijetu (npr. o tlaku, temperaturi, brzini vrtnje, itd.), npr. primamo podatke o procesu kojim upravljamo (u sebi sadrži AD pretvornik)
  - Relej kojim uključujemo ili isključujemo neki uređaj kojim upravljamo (npr. motor, klima uređaj, alarm)
  - Pisač na koji šaljemo tekst za ispis
  - Zaslon i tipkovnica preko kojih komunicira korisnik
  - itd.

Razmislite, prodiskutirajte,...

- Da li možemo spojiti npr LED ili relej direktno na podatkovnu liniju našeg procesora ?
- Da li možemo spojiti neku tipku/prekidač direktno na podatkovnu liniju ?



Povezivanje računala s okolinom

- Podsjetnik: naš projekt

Povezivanje računala s okolinom

- Na koji način se konkretno povezuju ovakvi ili bilo koji drugi uređaji s računalom?
- Kako je najpraktičnije komunicirati s ovakvim uređajima?
- Kako se komunicira s UI jedinicama i kako ih se adresira?
- itd...

Komunikacija procesora s UI jedinicama

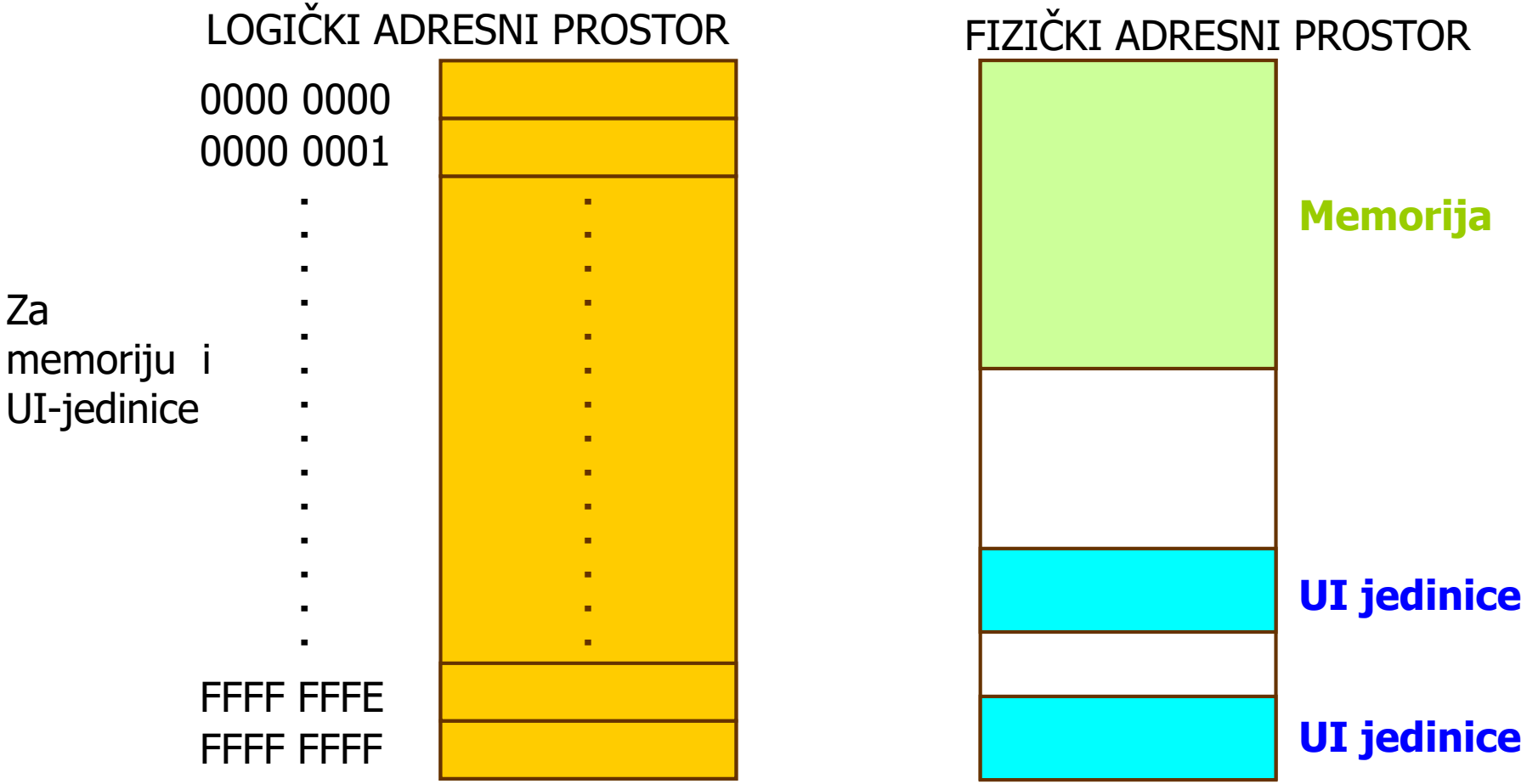
Načini adresiranja UI jedinica

- Načelno, **kod UI komunikacije postoje dvije operacije** kao i kod pristupanja memoriji:
  - čitanje
  - pisanje

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

- Dva osnovna načina adresiranja UI jedinica su:
  - **memorijsko UI preslikavanje** (memory mapped IO)
  - **izdvojeno UI adresiranje** (isolated IO)

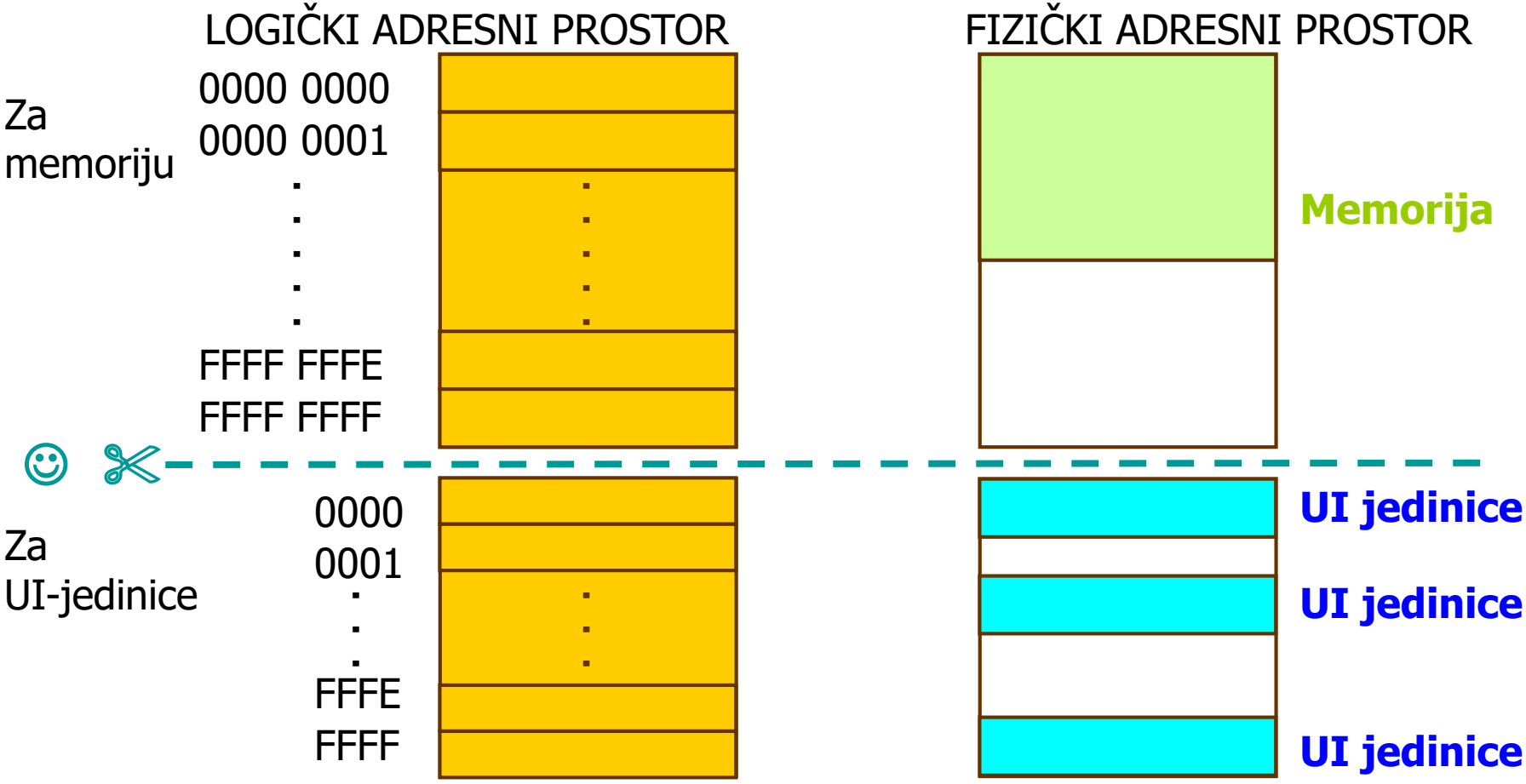
Memorijsko UI preslikavanje



Memorijsko UI preslikavanje

- **Memorijsko UI preslikavanje** ima sljedeće značajke:
  - procesor na jednak način pristupa memorijskim lokacijama i UI jedinicama
  - sabirnički protokoli čitanja i pisanja jednaki su za pristup memoriji i UI jedinicama

Izdvojeno UI adresiranje



Izdvojeno UI adresiranje

- **Izdvojeno UI adresiranje** ima sljedeće značajke
  - procesor na različite načine pristupa memoriji i UI jedinicama.
  - procesor ima posebne priključke kojima određuje je li neka adresa (npr. 10), koja se nalazi na adresnoj sabirnici, namijenjena memoriji ili UI jedinici

Usporedba

- Prednosti i nedostaci memorijskog i izdvojenog adresiranja:

Memorijsko UI preslikavanje

- jedan protokol
- bez dodatnih priključaka
- bez dodatnih naredaba
- manji adresni prostor i miješanje adresa

Izdvojeno UI adresiranje

- više sabirničkih protokola
- dodatni priključci
- dodatne naredbe (npr. IN i OUT)
- puni adresni prostor bez miješanja

Komunikacija FRISC-a s UI jedinicama

- **Za FRISC ćemo odabrati memorijsko preslikavanje**:
  - bolje je prilagođeno RISC procesorima zbog veće jednostavnosti (samo jedan sabirnički protokol i nepotrebne dodatne naredbe)
  - memorijski adresni prostor je više nego dovoljan za naše potrebe pa možemo jedan njegov dio odvojiti za UI jedinice
  - komunikacija s UI jedinicama odvija se pomoću naredaba LOAD i STORE budući da su to jedine naredbe koje pristupaju memoriji
    - umjesto adresa memorijskih lokacija, u naredbama LOAD i STORE koristit će se adrese UI jedinica
    - **u zadatcima i na labosima po dogovoru** uzimamo da je na FRISC spojena memorija na adresama 0 do FFFF (64 kB), a UI jedinice nalazit će se na adresama FFFF0000 do FFFFFFFF (sve možemo dohvatiti 20-bitnom adresom)

Vrste UI prijenosa podataka

- UI jedinice rade svojom brzinom koja je **različita** (obično manja) od brzine procesora

- Prije prenošenja podataka, obično se procesor i UI jedinica trebaju **sinkronizirati**, tj. uskladiti svoj rad da bi se prijenos podataka uspješno izveo

- S obzirom na to, prijenos se može dijeliti na:
  - **programski prijenos**
  - **sklopovski prijenos**

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**
  - U nastavku ćemo vidjeti tri vrste programiranog prijenosa:
    - **bezuvjetni**
    - **uvjetni**
    - **prekidni**

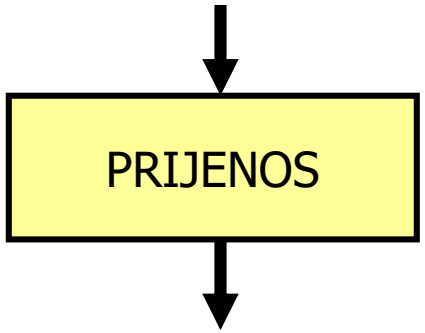
Sklopovski prijenos

- 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 se prenose izravno između UI 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
  - na kraju ovog poglavlja ćemo vidjeti jednu vrstu sklopovskog prijenosa:
    - **izravni pristup memoriji (DMA)**

Bezuvjetni prijenos

Bezuvjetni prijenos

- **Bezuvjetni prijenos** je najjednostavnija vrsta prijenosa
- Glavna značajka je da se prije prijenosa **ne provjerava** je li UI jedinica spremna za prijenos podataka
  - nema sinkronizacije brzine rada između procesora i VJ
- Dijagram toka je trivijalan i sastoji se samo od prijenosa:



- Prijenos je jedna naredba (eventualno više njih) za čitanje iz VJ ili za pisanje u VJ

Bezuvjetni prijenos

- UI jedinica je sklopovski **najjednostavnija**
- **Najbrži** prijenos među programiranim prijenosima
- **Nedostatak je mogućnost da UI jedinica nije spremna za prijenos…**

Bezuvjetni prijenos

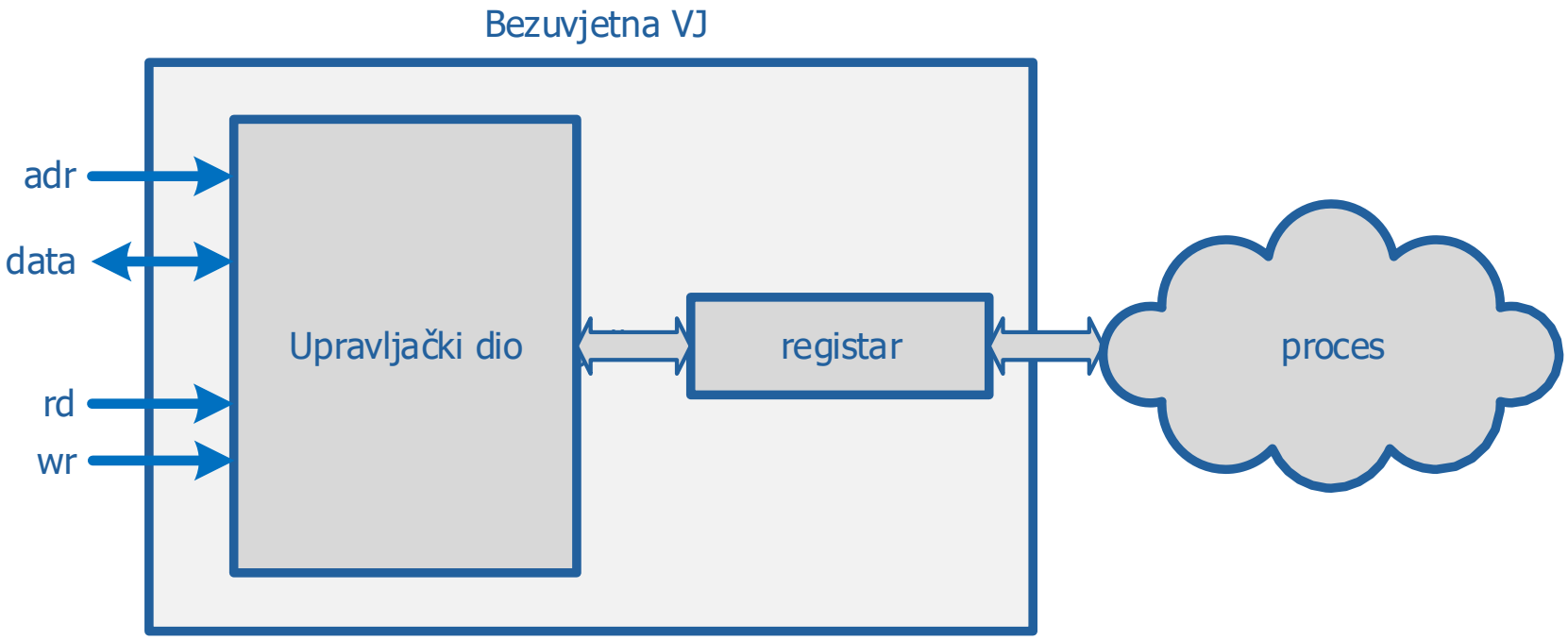
- Kad možemo koristiti bezuvjetni prijenos?

- Općenito:
  - kad nam nije bitna sinkronizacija
  - kad znamo da nećemo pristupati UI jedinici brže nego što ona radi

- Na primjer: želimo očitati trenutačnu vrijednost temperature u prostoriji preko temperaturnog senzora

- Na primjer: želimo svake sekunde osvježiti prikaz vremena na zaslonu (npr. na 7-segmentnom LCD-u)

Bezuvjetna VJ

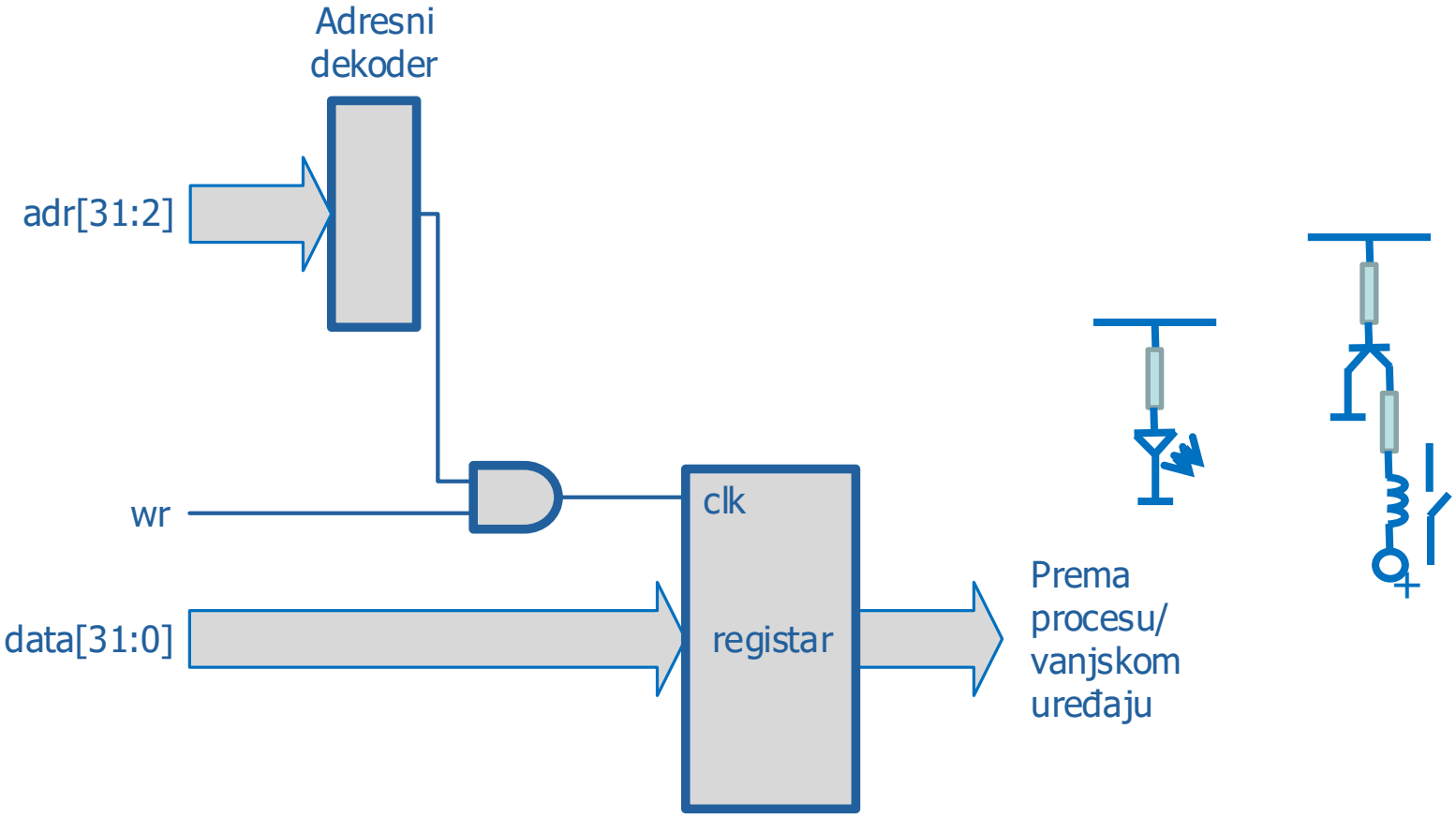


Osnovna građa bezuvjetne VJ

- Sve naše VJ zauzimat će određeni broj uzastopnih lokacija (jednu ili više njih)
- Zbog jednostavnosti, sve ove lokacije će biti 32-bitne pa ćemo s VJ komunicirati korištenjem naredaba LOAD i STORE
  - Dakle, svaka lokacija će zauzimati 4 adrese
- Bezuvjetna VJ zauzima samo jednu lokaciju

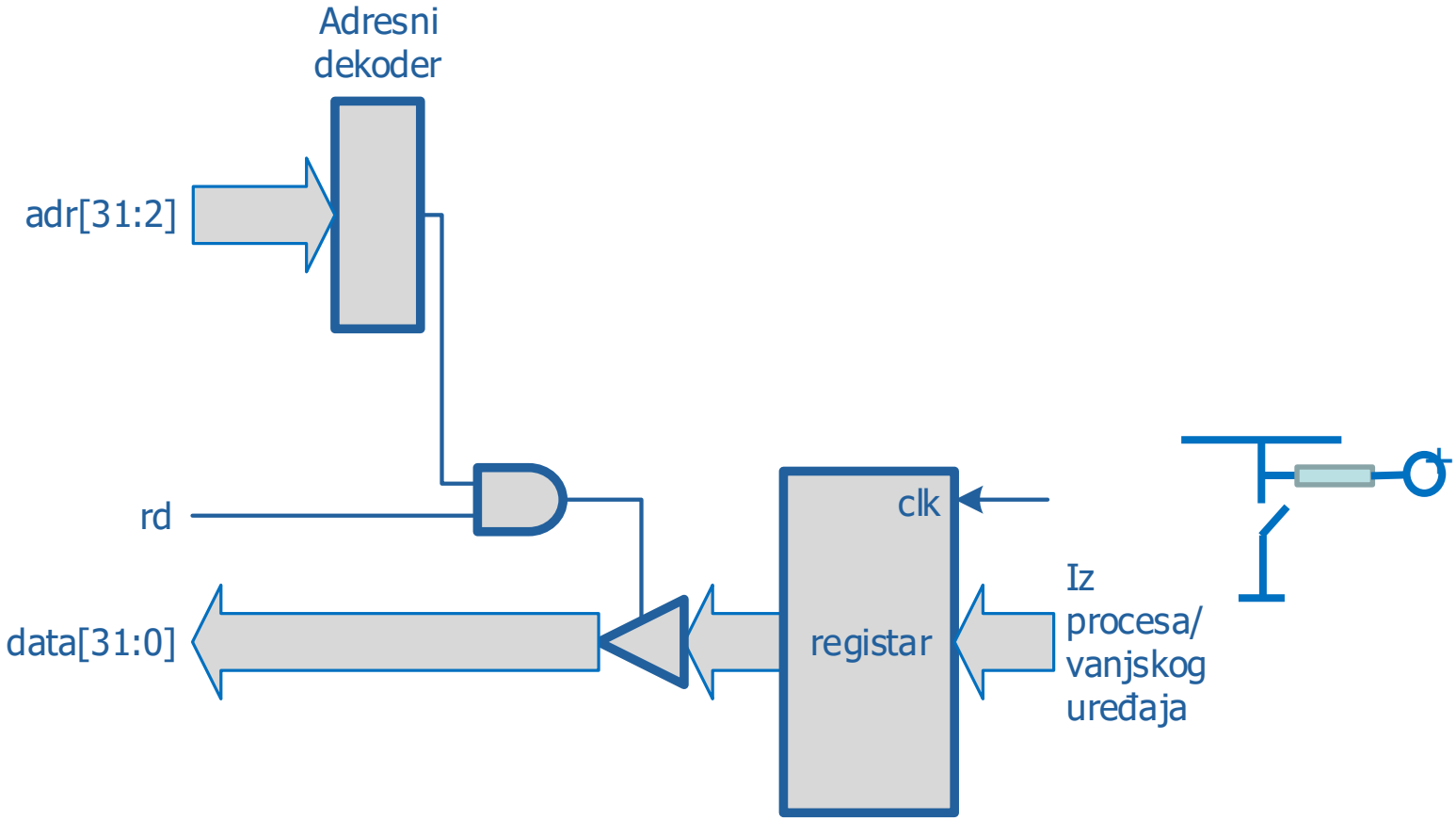
Bezuvjetna VJ

- Arhitektura jednostavne izvedbe bezuvjetne **izlazne VJ**



Bezuvjetna VJ

- Arhitektura jednostavne izvedbe bezuvjetne **ulazne VJ**



Bezuvjetni prijenos - Primjeri

Na bezuvjetnu izlaznu VJ0 na adresi FFFF 0000 spojen je relej. Slanjem broja 0 relej se isključuje, a slanjem 1 se uključuje. Na bezuvjetnu ulaznu VJ1 na adresi FFFF 0010 spojena je tipka. Kad je tipka pritisnuta, s VJ se očitava stanje 1, a kad tipka nije pritisnuta, očitava se stanje 0.

Napisati program koji treba ispitivati je li tipka pritisnuta i samo tada držati relej uključen. Građa VJ0 i VJ1 je kao u prethodnom opisu.

Bezuvjetni prijenos - Primjeri

```
RELEJ EQU  0FFFF0000   ; Definiranje naziva VJ
TIPKA EQU  0FFFF0010

;  Glavni program
PETLJA   LOAD  R0, (TIPKA)
         OR    R0, R0, R0
         JR_Z  ISKLJUCI

UKLJUCI  MOVE  1, R0
         STORE R0, (RELEJ)
         JR    PETLJA

ISKLJUCI MOVE  0, R0
         STORE R0, (RELEJ)
         JR    PETLJA
```

Bezuvjetni prijenos - Primjeri

Na bezuvjetnu ulaznu vanjsku jedinicu na adresi FFFF 0000 spojen je digitalni termometar s kojeg se može očitati trenutna vrijednost temperature. Vanjska jedinica građena je kao što je pokazano na prethodnim slikama.

Program mora svakih 5 minuta očitati temperaturu, izračunati srednju vrijednost svih dotadašnjih temperatura i spremiti je u lokaciju SR_TEMP. Nakon tri sata treba zaustaviti procesor. Pretpostavite da već postoji potprogram za dijeljenje (DIJELI) kojem su parametri u fiksnim memorijskim lokacijama SUMA i BROJ_OCITANJA, a rezultat se vraća registrom R0.

Pretpostavka je da se vrijednost temperature vraća u nižih 8 bitova, a gornjih 24 bita su nule, te da procesor radi sa signalom vremenskog vođenja *clock* frekvencije 10 MHz.

Bezuvjetni prijenos - Primjeri

```
;  Definiranje adrese vanjske jedinice
TERMOMET EQU  0FFFF0000

;  Glavni program
        MOVE   10000, R7

        ; Inicijalizacija varijabli
        MOVE   0, R0
        STORE  R0, (SUMA)
        STORE  R0, (BROJ_OCITANJA)
        STORE  R0, (SR_TEMP)
        MOVE   %D 36, R1  ; brojač za 3 sata: 36*5 min = 3 sata

PETLJA  LOAD  R0, (TERMOMET) ; Učitaj temperaturu
        CALL  RACUNAJ_SREDNJU ;Izračunaj i spremi
        STORE R0, (SR_TEMP)   ;srednju temperat.

        CALL  KASNI_5_MINUTA

        SUB   R1, 1, R1      ; Istekla 3 sata?
        JR_NZ PETLJA

        HALT
```

Bezuvjetni prijenos - Primjeri

```
        ; Potprogram za izračun sredine:
RACUNAJ_SREDNJU ; Parametar R0: nova temperatura
                ; Rezultat u R0

        PUSH  R1

        LOAD  R1, (SUMA)
        ADD   R0, R1, R1
        STORE R1, (SUMA)

        LOAD  R1, (BROJ_OCITANJA)
        ADD   R1, 1, R1
        STORE R1, (BROJ_OCITANJA)

        CALL  DIJELI

        POP   R1
        RET
```

Bezuvjetni prijenos - Primjeri

```
        ; Potprogram za (približno) 5-minutno kašnjenje
        ; Nema parametara ni rezultata

KASNI_5_MINUTA
       PUSH  R0
       LOAD  R0, (KONST)     ; 10^9 prolazaka

LOOP   SUB   R0, 1, R0       ; 1 takt
       JR_NZ LOOP            ; 2 takta
       POP   R0
       RET

KONST  DW  %D 1000000000

; Ovaj potprogram napisan je za CLOCK frekvencije 10 MHz:
; 5 min = 300 s = 300*10*10^6 taktova  = 3*10^9 taktova


; Varijable
SUMA            DW 0
BROJ_OCITANJA   DW 0
SR_TEMP         DW 0
```

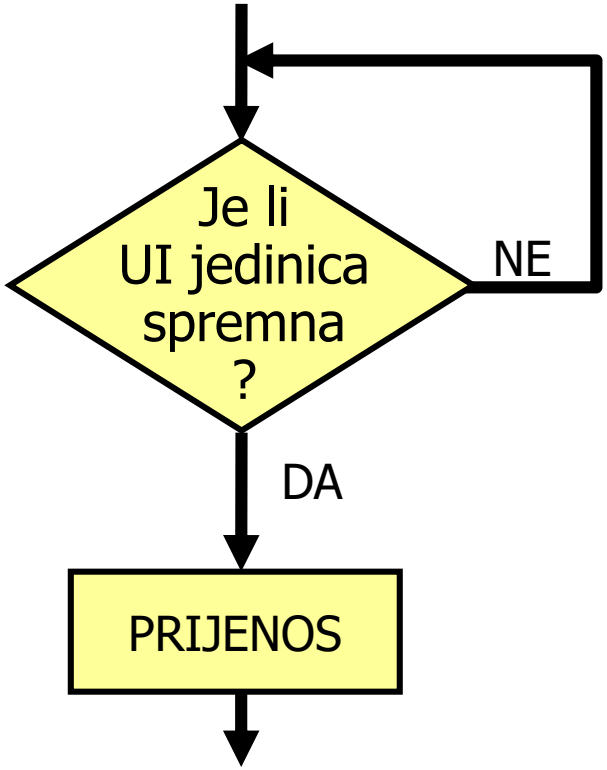
Bezuvjetni prijenos - Primjeri

- DZ: Proučiti dodatne primjere iz knjige i zbrike

Uvjetni prijenos

Uvjetni prijenos

- **Uvjetni prijenos** rješava probleme gubitka i uvišestručenja podataka kod bezuvjetnog prijenosa
- Glavna značajka je da se prije prijenosa **uvijek provjerava** je li VJ spremna za prijenos podataka



Uvjetni prijenos

- Glavni **nedostatak** uvjetnog prijenosa:
  - Budući da je procesor tipično puno brži od vanjskih uređaja, može se dogoditi da procesor puno vremena gubi na čekanje da VJ postane spremna

- Uvjetna VJ je sklopovski složenija od bezuvjetne VJ:
  - VJ mora imati stanje spremnosti koje se pamti u **bistabilu stanja** (u tzv. status-bistabilu)
  - VJ mora imati dodatne **sinkronizacijske priključke** za povezivanje s vanjskim uređajem

Uvjetni prijenos

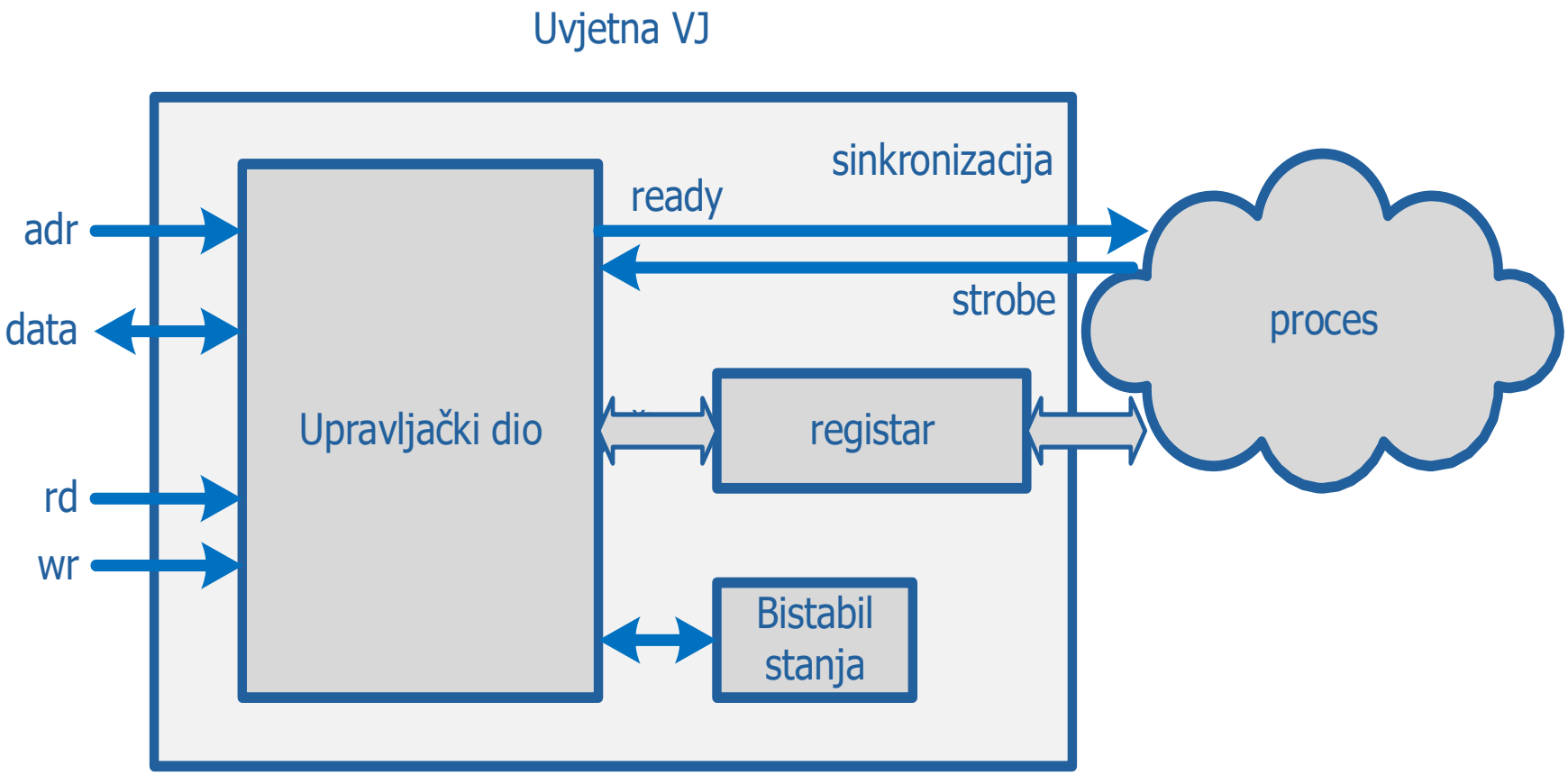
- Kad možemo koristiti uvjetni prijenos?

- Općenito:
  - kad nam je bitno da nema gubitaka/uvišestručenja podataka (tj. bitna nam je sinkronizacija)
  - kad ne znamo kojom brzinom ćemo pristupati VJ pa se može dogoditi da ona još nije spremna

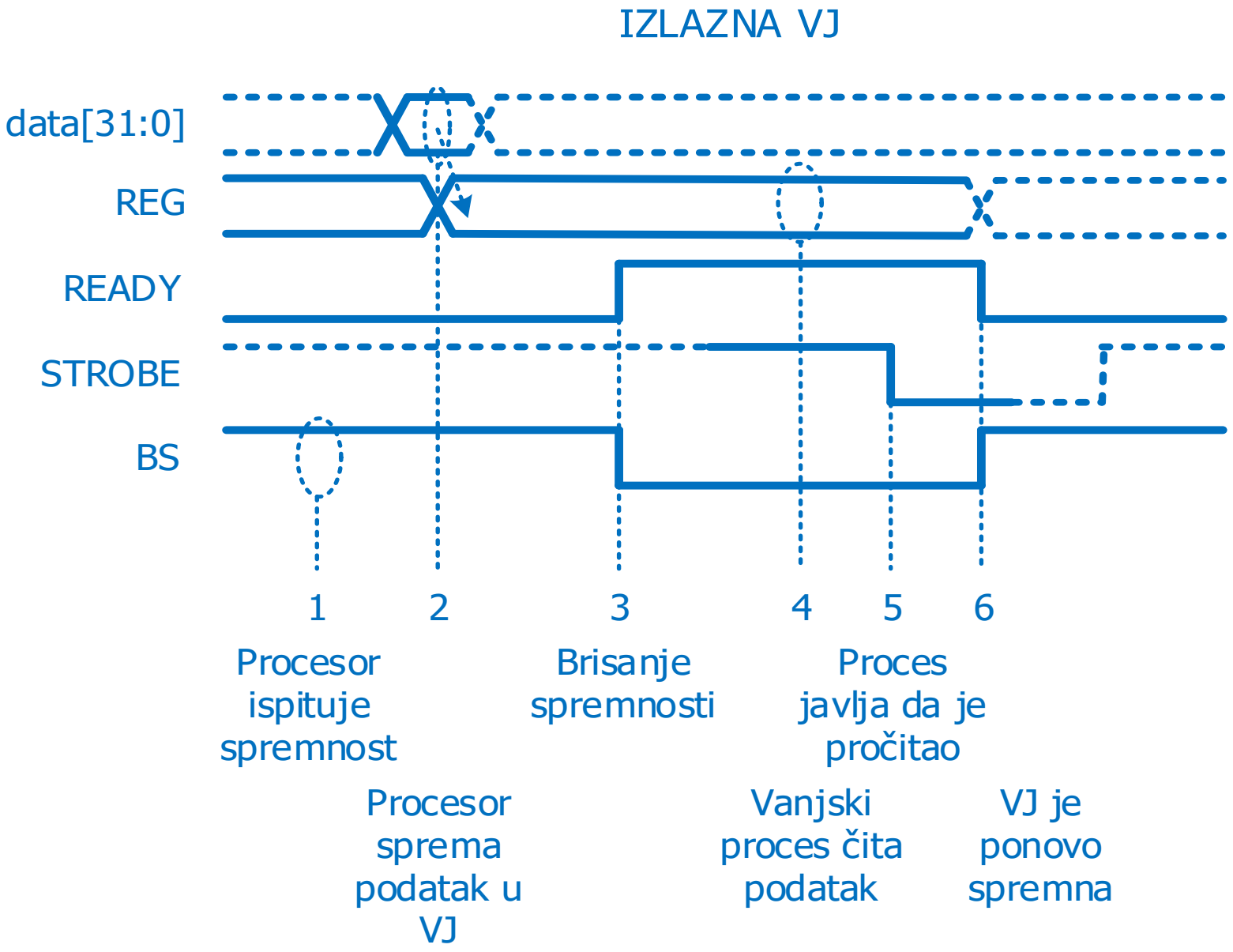
- Na primjer: želimo slati znakove na pisač (naravno, pri tome je bitno da svi znakovi budu primljeni i ispisani)

- Na primjer: želimo čitati niz podataka koji od nekuda (npr. s mreže ili iz tipkovnice) pristižu na VJ i koje sve želimo bez gubitaka pohraniti u memoriju

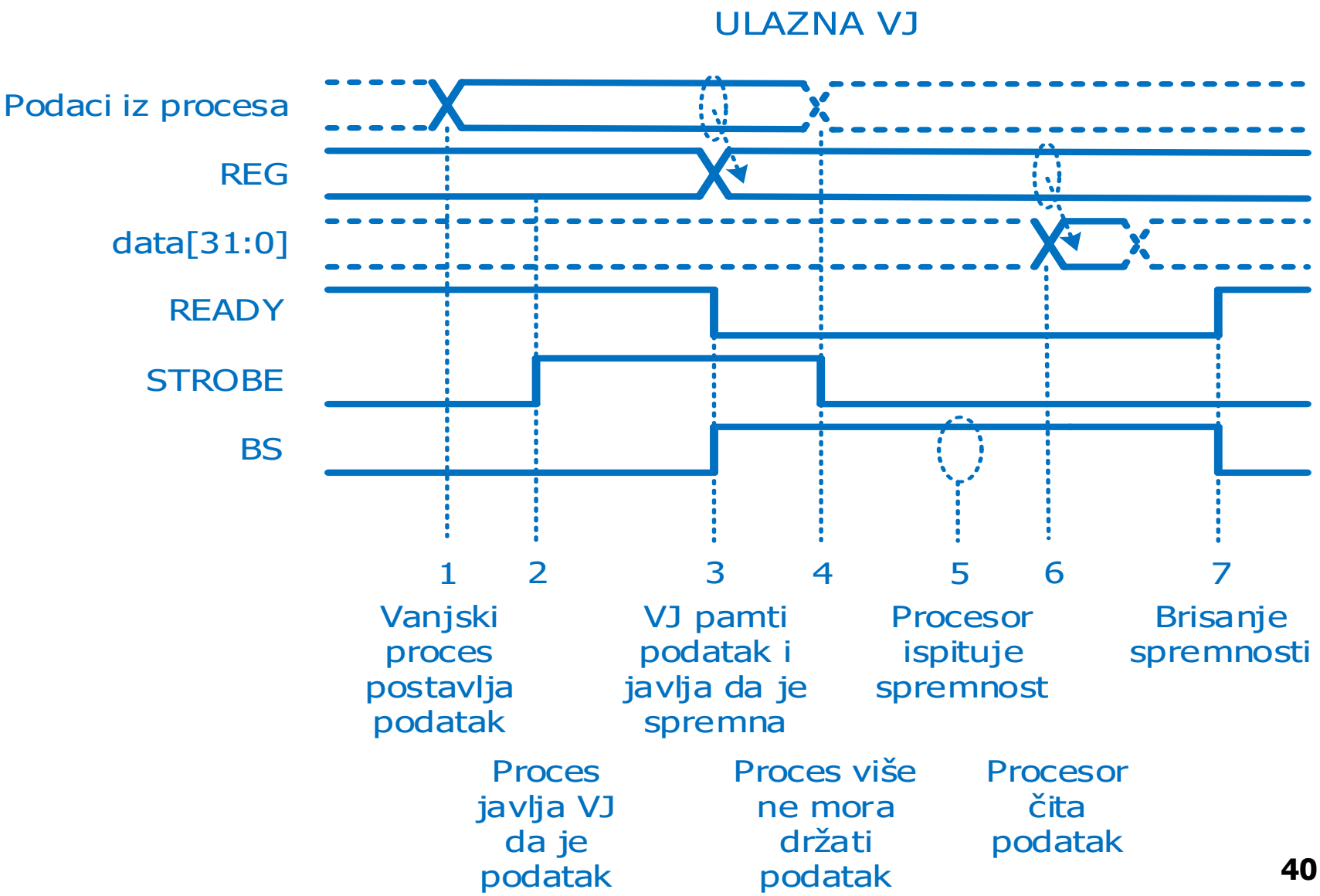
Osnovna građa uvjetne UI jedinice



Vremenski dijagram uvjetne izlazne komunikacije



Vremenski dijagram uvjetne ulazne komunikacije



Uvjetni prijenos - rekapitulacija

- Iz prethodnih opisa vidi se da uvijek vrijedi:
  - Dok je VJ spremna za komunikaciju s procesorom, nije spremna za komunikaciju s vanjskim procesom (i obrnuto)
- Bistabil stanja služi za sinkronizaciju između VJ i procesora:
  - Procesor programski ispituje b.stanja da bi utvrdio spremnost VJ
  - VJ postavlja bistabil stanja kad postane spremna
  - Bistabil stanja briše se (programski ili automatski) nakon obavljenog prijenosa
  - **Brisanjem bistabila stanja, omogućuje se nastavak komunikacije s vanjskim procesom**
- Sinkronizacijski priključci služe za sinkronizaciju između VJ i vanjskog procesa:
  - Vanjski proces sklopovski ispituje sinkronizacijske priključke da bi utvrdio spremnost VJ

Osnovna građa uvjetne UI jedinice

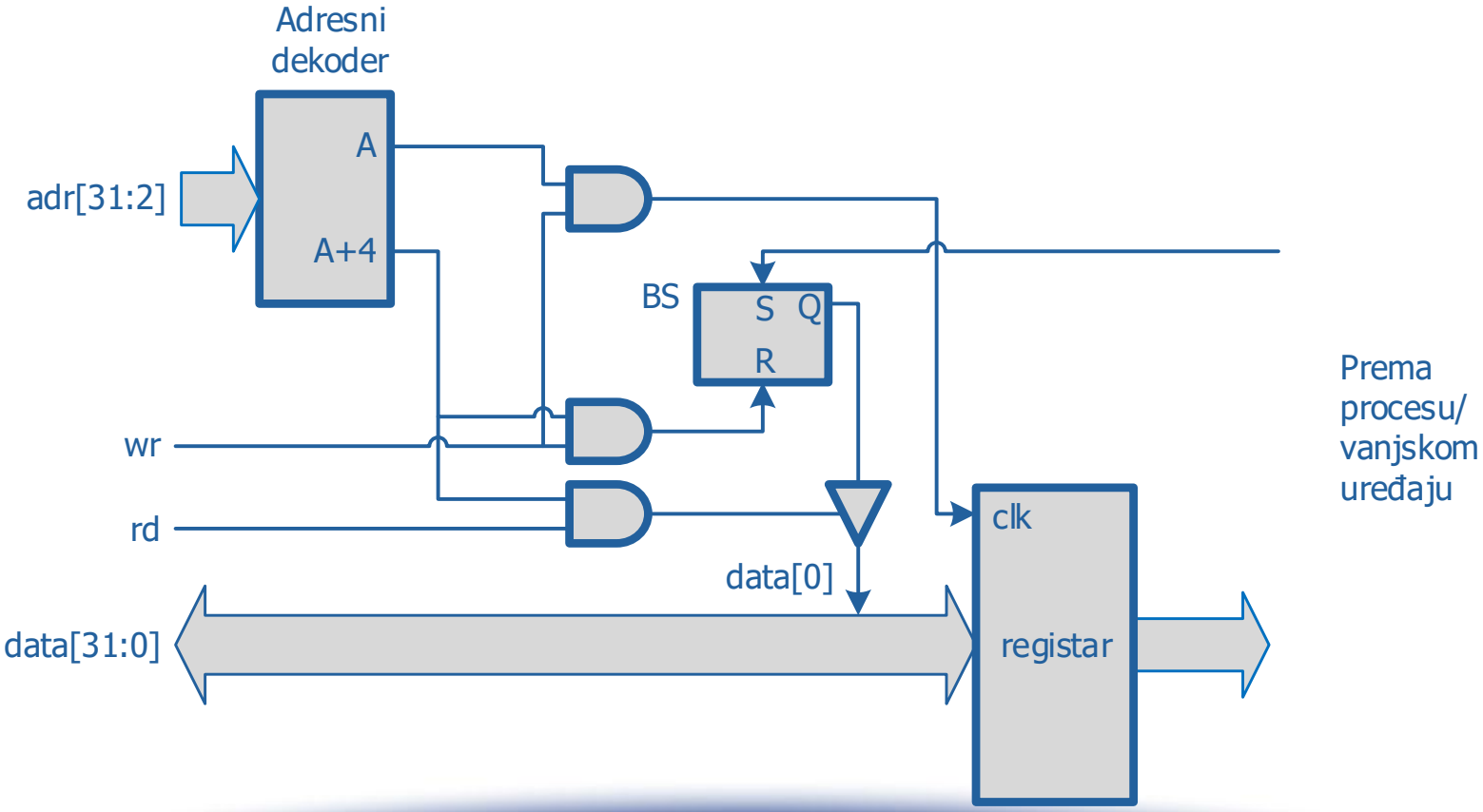
- Ovakva vanjska jedinica **zauzimat će dvije uzastopne 32-bitne lokacije**
  - Na prvoj lokaciji se čita ili piše podatak
  - Na drugoj lokaciji se pristupa bistabilu stanja

- Na lokaciji za bistabil stanja može se:
  - Pročitati trenutačni sadržaj bistabila (ispitivanje stanja)
  - Obrisati bistabil (operacijom upisa bilo kojeg podatka pri čemu se poslani podatak zanemaruje) *

* Za naše uvjetne UI jedinice ćemo pretpostavljati da se bistabil stanja uvijek mora brisati programski (tj. pretpostavljamo da se to ne radi automatski)

Arhitektura izlazne uvjetne VJ

- Pojednostavljeni prikaz upravljačkog dijela **izlazne uvjetne VJ**



Uvjetni prijenos - Primjeri

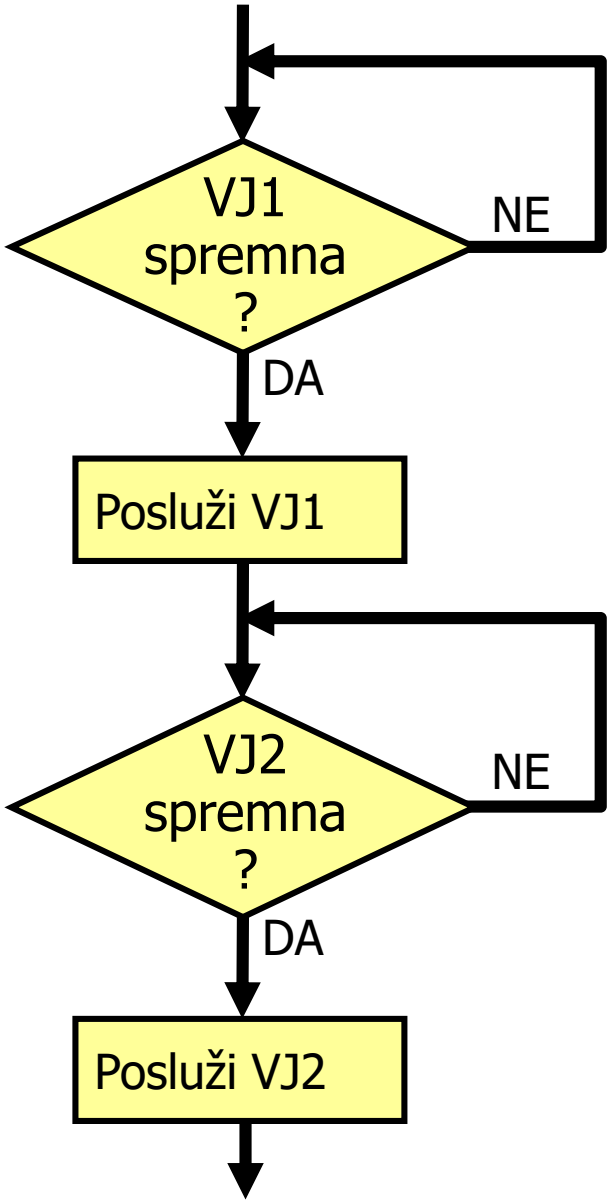
Napišite program koji će na uvjetnu vanjsku jedinicu (građenu kao što je upravo opisano) koja se nalazi na adresi 0FFFF0000 poslati 200 32-bitnih podataka koji se nalaze u memoriji od lokacije PODACI.

Nakon što su svi podaci poslani, treba zaustaviti procesor.

```
;  Definiranje adresa vanjske jedinice
REG     EQU     0FFFF0000
BS      EQU     0FFFF0004

;  Glavni program
        ; Inicijalizacija varijabli
        MOVE  PODACI, R0          ; adresa podataka
        MOVE  %D 200,  R1         ; brojač petlje

CEKAJ   LOAD  R2, (BS)            ; čitaj stanje iz VJ
        OR    R2, R2, R2          ; postavi zastavice
        JR_Z  CEKAJ               ; ispitaj stanje VJ

SPREMNA LOAD  R2, (R0)            ; čitaj podatak iz mem.
        STORE R2, (REG)           ; šalji podatak u VJ
        STORE R2, (BS)            ; briši stanje VJ

        ADD   R0, 4, R0
        SUB   R1, 1, R1
        JR_NZ CEKAJ
        HALT

PODACI DW    12, 5452, 331A3, ...
```

Posluživanje više uvjetnih vanjskih jedinica

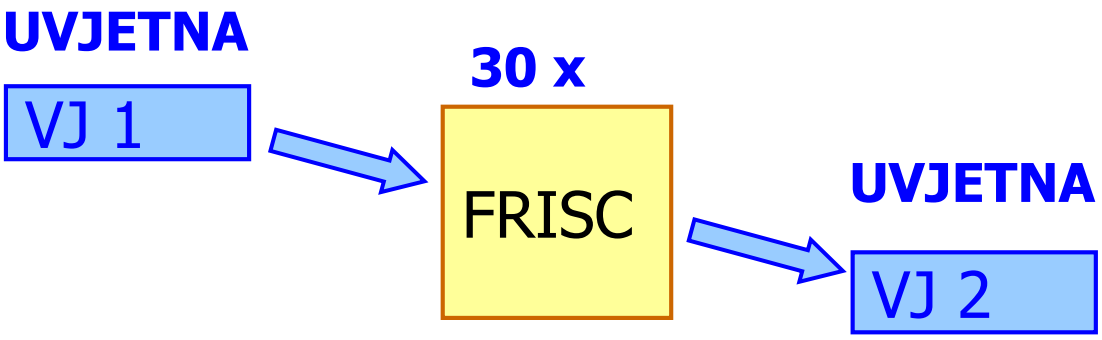
- Ako su **jedinice zavisne**, onda moramo za svaku **čekati** da postane spremna
- Zavisne jedinice znače da je redoslijed njihovog posluživanja bitan
- Na slici je primjer gdje se podatak primljen od VJ1 prenosi na VJ2



Uvjetni prijenos – dvije VJ

Na FRISC su spojene dvije uvjetne vanjske jedinice: ulazna vj1 na adresi FFFF1000 te izlazna vj2 adresi FFFF2000.

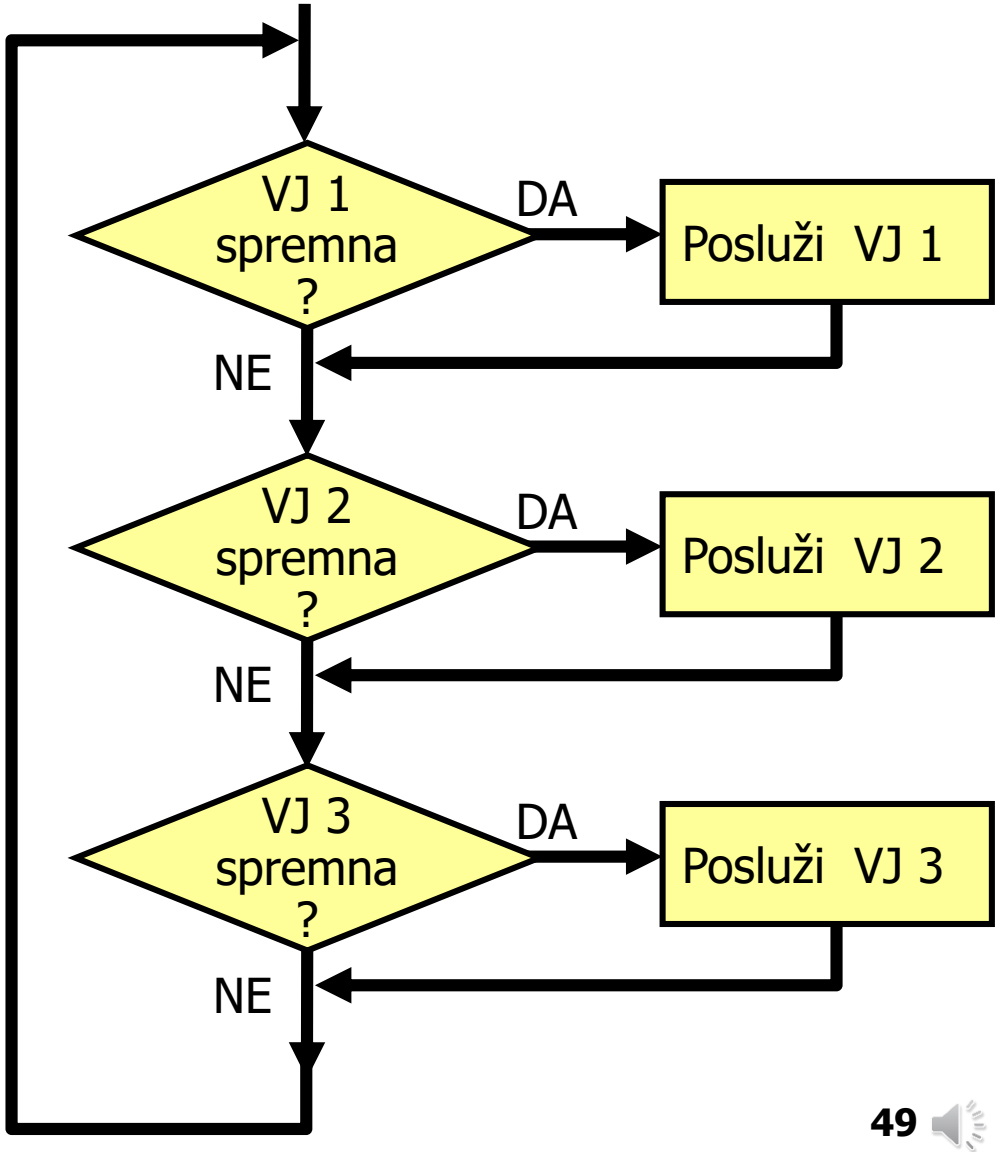
FRISC treba prenijeti 30 podataka sa VJ1 na VJ2 nakon čega nastavlja s izvođenjem glavnog programa.



```
;  Definiranje adresa vanjskih jedinica
PRIMI_1  EQU   0FFFF1000
BS_1     EQU   0FFFF1004
SALJI_2  EQU   0FFFF2000
BS_2     EQU   0FFFF2004

; Glavni program
         MOVE   30,  R3           ; brojač podataka

CEKAJ_1  LOAD    R0, (BS_1)
         OR      R0, R0, R0
         JR_Z    CEKAJ_1          ; Čekaj da VJ1 postane spremna
         LOAD    R1, (PRIMI_1)    ; Posluži VJ1
         STORE   R0, (BS_1)

CEKAJ_2  LOAD    R0, (BS_2)
         OR      R0, R0, R0
         JR_Z    CEKAJ_2          ; Čekaj da VJ2 postane spremna
         STORE   R1, (SALJI_2)    ; Posluži VJ2
         STORE   R0, (BS_2)

         SUB     R3, 1, R3
         JR_NZ   CEKAJ_1          ; Ispitaj je li preneseno svih 30 podataka
         . . .
```

Posluživanje više uvjetnih vanjskih jedinica

- Ako ima **više nezavisnih uvjetnih VJ**, onda se može primijeniti postupak **prozivanja** (eng. polling)
- Nezavisne jedinice znače da jedna ne ovisi o drugoj, tj. da redoslijed posluživanja nije bitan
- **Prozivanje umanjuje nedostatak uvjetnog prijenosa**:
  - Čekanje na spremnost pojedine vanjske jedinice
  - Veća je vjerojatnost da će jedna od nekoliko jedinica postati spremna
- Nakon posluživanja jedne jedinice prelazi se na sljedeću jedinicu
  - Ne bi bilo dobro vratiti se na ispitivanje prve zbog mogućeg "izgladnjivanja" zadnjih jedinica u lancu ispitivanja

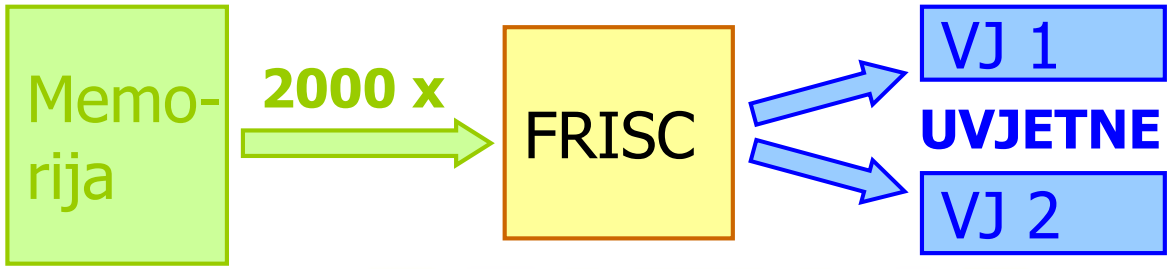


Uvjetni prijenos – primjer

Na FRISC su spojene dvije uvjetne vanjske jedinice: vj1 na adresi FFFF1000 i vj2 adresi FFFF2000.

Obje jedinice rade neovisno i FRISC im šalje 2000 podataka smještenih od adrese PODACI. Kako koja jedinica postane spremna tako joj se pošalje podatak (to znači da će brža jedinica primiti više podataka).

Kada je završen prijenos svih 2000 podataka, treba zaustaviti procesor.



```
;  Definiranje adresa vanjskih jedinica
SALJI_1   EQU   0FFFF1000
BS_1      EQU   0FFFF1004
SALJI_2   EQU   0FFFF2000
BS_2      EQU   0FFFF2004

; Glavni program
          MOVE   10000, R7
          MOVE   2000,  R1            ; brojač podataka
          MOVE   PODACI, R2           ; adresa podataka

; POSTUPAK PROZIVANJA
PROZIVAJ  LOAD    R0, (BS_1)
          OR      R0, R0, R0
          CALL_NZ SALJI_VJ1

          OR      R1, R1, R1          ; ispitaj uvjet za kraj (2000 podataka)
          JR_Z    KRAJ

          LOAD    R0, (BS_2)
          OR      R0, R0, R0
          CALL_NZ SALJI_VJ2

          OR      R1, R1, R1          ; ispitaj uvjet za kraj (2000 podataka)
          JR_NZ   PROZIVAJ

KRAJ      HALT
```

```
; potprogram za posluživanje vj1

SALJI_VJ1 LOAD  R0, (R2)            ; podatak iz memorije
          STORE R0, (SALJI_1)       ; šalji podatak
          STORE R0, (BS_1)          ; briši spremnost
          ADD   R2, 4, R2           ; pomakni pokazivač
          SUB   R1, 1, R1           ; smanji brojač
          RET



; potprogram za posluživanje vj2 (isto kao i za vj1)

SALJI_VJ2 LOAD  R0, (R2)
          STORE R0, (SALJI_2)
          STORE R0, (BS_2)
          ADD   R2, 4, R2
          SUB   R1, 1, R1
          RET
```

Posluživanje više VJ - Komentar

• Načelno, kod posluživanja više vanjskih jedinica treba odrediti redoslijede čitanja/pisanja u ovisnosti o samim jedinicama:

- **Uvjetnim VJ** treba uvijek ispitati spremnost i poslužiti ih čim se ustanovi da su spremne
- Ako su uvjetne VJ nezavisne, onda ih se proziva
- Ako su uvjetne VJ zavisne, onda se mora čekati da postanu spremne

- **Bezuvjetne VJ** treba čitati/pisati tek kad zatreba podatak
  - Nema smisla npr. pročitati podatak s bezuvjetne, a onda (dugo) čekati da uvjetna postane spremna kako bi joj poslali taj podatak
  - Treba pokušati raditi s "najsvježijim" podatcima: npr. čekati da uvjetna postane spremna i tek onda pročitati podatak s bezuvjetne i odmah ga poslati uvjetnoj

Uvjetni prijenos - Primjeri

- DZ: Proučiti dodatne primjere iz knjige i zbirke