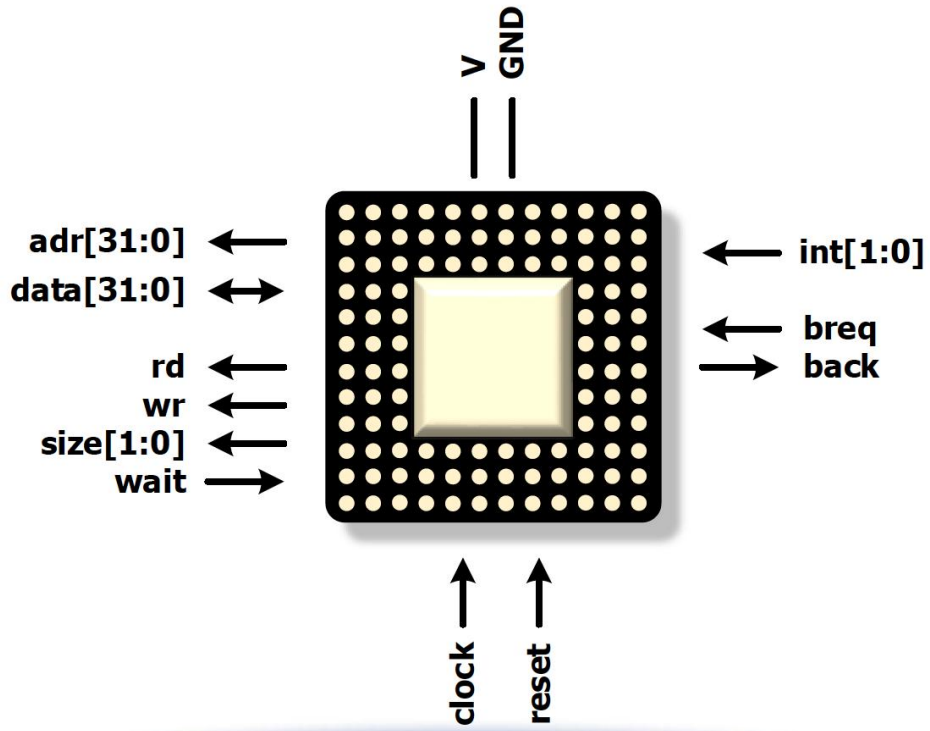
Priključci - podjela po namjeni

- Sabirnice se prema namjeni obično dijele na:
  - adresnu sabirnicu (address bus)
  - podatkovnu sabirnicu (data bus)
  - upravljačku sabirnicu (control bus)
- Adresni priključci postavljaju adresu na adresnu sabirnicu, od procesora prema memoriji i vanjskim jedinicama. Procesor, kao aktivna i vodeća komponenta, adresira druge komponente zadajući im adresu s koje želi čitati ili pisati podatak
- Podatkovni priključci spojeni su na podatkovnu sabirnicu i služe za prijenos podataka između procesora i memorije ili između procesora i vanjske jedinice prilikom operacija čitanja i pisanja
- Upravljački priključci imaju razne funkcije vezane uz rad procesora i općenito služe za sinkronizaciju rada pojedinih dijelova računala

Priključci - smjer priključaka

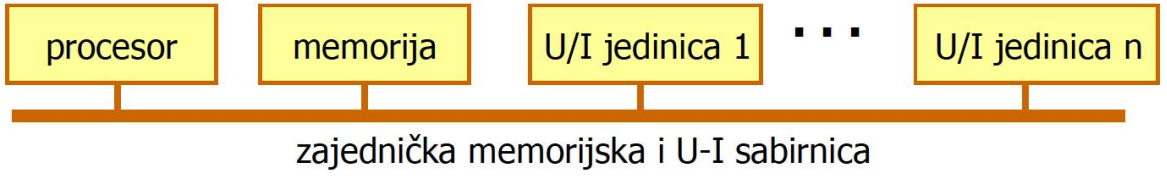
- Priključci se mogu podijeliti po smjeru signala (podataka) koji njima putuje na:
  - ulazne
  - izlazne
  - dvosmjerne
- Smjer priključka uvijek se definira u odnosu na komponentu koju promatramo

Priključci procesora FRISC

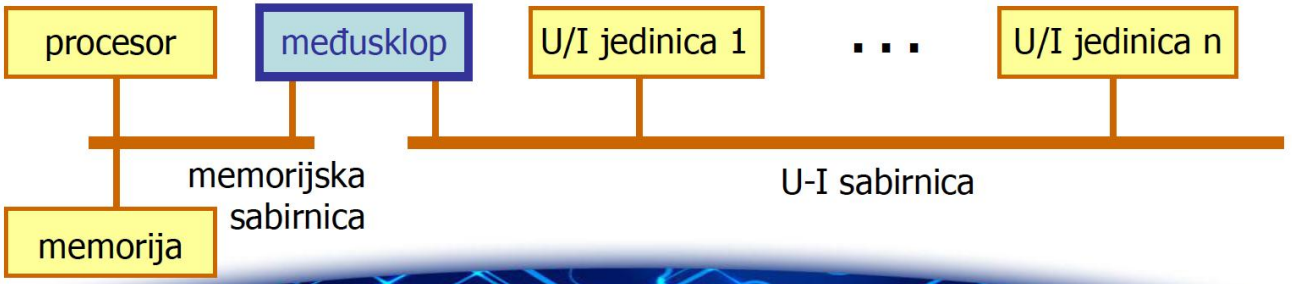


Sabirnice - memorijske i UI

- **Zajednička** memorijska i U-I sabirnica (backplane bus)



- Spajanje memorijske i UI sabirnice pomoću posebnog **međusklopa** (tj. neizravno)



Sabirnice - sinkrone i asinkrone

- Sabirnice se prema načinu komunikacije dijele na:
  - sinkrone
  - asinkrone

- **Sinkrone sabirnice:**
  - **sve operacije su sinkronizirane s taktom sustava (tj. clockom)**
  - jednostavne su za implementaciju
  - imaju veliku brzinu rada pa zato i malu duljinu
  - bolje su prilagođene za slučaj kad svi uređaji imaju jednaku brzinu
  - imaju mogućnost prilagodbe brzine rada, ali se komunikacija većinom odvija predviđenom brzinom
  - češće se koriste za memorijske sabirnice

Sabirnice - asinkrone

- **Asinkrone sabirnice:**
  - ne koriste CLOCK za sinkronizaciju
  - **uređaji se sinkroniziraju tzv. rukovanjem** (engl. handshaking protocol)
    - rukovanje je postupak u kojem strane koje komuniciraju prelaze na sljedeći korak komunikacije tek kad obje strane potvrde da je prethodni korak dovršen
  - složenije su za implementaciju
  - imaju manju brzinu rada
  - mogu imati veliku duljinu
  - bolje su prilagođene za slučaj kad uređaji imaju različite brzine
  - češće se koriste za U-I sabirnice

Sabirnice - sinkrone i asinkrone

- Komentari:
  - I sinkrone i asinkrone sabirnice imaju mogućnost prilagodbe različitim brzinama uređaja spojenih na sabirnicu. Osnovna razlika je:
    - u tome što su asinkrone sabirnice upravo predviđene za spajanje uređaja različite brzine
    - spajanje uređaja različite brzine je više iznimka nego pravilo kod sinkronih sabirnica, ili takvi uređaji sudjeluju u manjem postotku sabirničkih transakcija
  - Asinkroni protokoli trebaju potvrdu da bi mogli nastaviti sa sljedećim koracima bez obzira koliko je vremena proteklo
  - Sinkroni protokoli nastavljaju s radom automatski kad se dosegne određeno stanje CLOCK-a, a usporenje rada se mora izričito zahtijevati

Sabirnice - FRISC

- Za FRISC odabiremo:
  - neće se koristiti posebni spojni putovi između raznih dijelova sustava
  - zajednička memorijska i U-I sabirnica (tzv. backplane)
  - sinkrona sabirnica s mogućnošću prilagodbe brzine
    - brzina se prilagođava umetanjem tzv. ciklusa čekanja

Sabirnice - FRISC

- Definirajmo sabirničke protokole za FRISC
  - Bit će jednaki za pristup memoriji ili U-I jedinicama (u objašnjenjima se spominje samo memorija, ali isto vrijedi i za U-I jedinice)
  - Pokazat ćemo protokole za čitanje i pisanje podataka
  - U slučaju normalne brzine, traju jedan takt CLOCK-a
  - U slučaju sporih memorija ili U-I jedinica, umeću se dodatni ciklusi čekanja (svaki traje po jedan takt CLOCK-a)
  - Promatramo samo one sabirničke vodove koji su relevantni za operacije čitanja i pisanja

Protočna struktura FRISC-a

- Kako bi zadržali jednostavnost našeg procesora, za efikasnije izvođenje naredaba uvesti ćemo najjednostavniju moguću protočnu strukturu
- Odabiremo **protočnu strukturu FRISC-a** sa samo **dvije razine**, koje ćemo nazvati:
  - **razina za dohvat**
  - **razina za izvođenje**

Protočna struktura FRISC-a

- Podjela operacija između razina je ovakva:
  - **Razina za dohvat**
    - Dohvat naredbe
    - Dekodiranje naredbe
    - Dohvat operanada
  - **Razina za izvođenje**
    - Izvođenje AL operacije
    - Spremanje rezultata

Podjela naredaba po brzini izvođenja

- Trajanje operacija u svakoj razini je jedan period signala vremenskog vođenja CLOCK-a (uz pretpostavku da memorija nije spora)
- Naredbe procesora FRISC razlikuju se po načinu kako se izvode:
  - **Faza dohvata svih naredaba traje jedan period**
  - **Faza izvođenja također traje jedan period, ali kod nekih naredaba nije moguće preklapanje s fazom dohvata sljedeće naredbe**

Jednociklusne naredbe

- Kod jednostavnih naredaba moguće je preklapanje faze izvođenja sa fazom dohvata sljedeće naredbe.
  - Ove naredbe zovu se **jednociklusne** jer im efektivno vrijeme izvođenja u protočnoj strukturi iznosi jedan period (ciklus) CLOCK-a.
  - Ove naredbe efikasno koriste protočnu strukturu.

- U jednociklusne naredbe spadaju:
  - Aritmetičko-logičke naredbe
  - Registarske naredbe
  - Upravljačke naredbe kod kojih uvjet nije zadovoljen

Jednociklusne naredbe

- Izvođenje slijeda jednociklusnih naredaba:

	Dohvat	Izvođenje
T1	ADD	
T2	SUB	ADD
T3	XOR	SUB
T4	AND	XOR
T5		AND

Dvociklusne naredbe

- Kod složenijih naredaba nije moguće preklapanje faze izvođenja s fazom dohvata sljedeće naredbe
  - Takve naredbe zovu se **dvociklusne** naredbe jer im efektivno vrijeme izvođenja u protočnoj strukturi iznosi dva perioda (ciklusa) CLOCK-a.
  - Ove naredbe ne koriste efikasno protočnu strukturu

- U dvociklusne naredbe spadaju:
  - Memorijske naredbe
  - Upravljačke naredbe kod kojih je uvjet za izvođenje zadovoljen

Dvociklusne naredbe

- Izvođenje slijeda dvociklusnih naredaba:

	Dohvat	Izvođenje
T1	LOAD	
T2		LOAD
T3	JP	
T4		JP
T5	STORE	
T6		STORE

Jednociklusne naredbe: dohvat

Prva polovica perioda:

- Postavljanje adrese na adresnu sabirnicu — PC → AR
- Aktiviranje signala rd **

Druga polovica perioda *:

- Čitanje naredbe — (AR) → IR
- Deaktiviranje signala rd **
- Dekodiranje naredbe — dekodiranje
- Dohvat operanada i slanje u ALU — operandi → ALU
- Izbor i pokretanje ALU operacije
- Postavljanje PC za sljedeću naredbu — PC+4 → PC

---

* Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran signal WAIT

** Aktiviranje i deaktiviranje signala rd i wr se neće više navoditi

Dvociklusne naredbe: dohvat

- Do trenutka dekodiranja sve naredbe imaju isti način dohvata naredbe

Jednociklusne naredbe: izvođenje

Prva polovica perioda:

- ALU završava operaciju i sprema se rezultat (osim kod CMP)
  ALU → Reg
- Spremaju se zastavice u SR

Druga polovica perioda:

- Nema aktivnosti vezano za izvođenje naredbe

Hazardi

- Do sada objašnjeni rad protočne strukture za jednociklusne naredbe bio je **idealan**, no u stvarnom radu postoje situacije koje uzrokuju da protočna struktura djelomično gubi svoju efikasnost
- U nekim situacijama protočna struktura ne može obraditi sljedeću naredbu odmah u sljedećem periodu
- Takve situacije nazivaju se **hazardi**
- Postoje tri osnovna tipa hazarda:
  - **Strukturni**
  - **Upravljački**
  - **Podatkovni**

Naredbe FRISC-a i strukturni hazard

- Zbog Von Neumannove arhitekture memorijskog sučelja, sve memorijske naredbe procesora FRISC (LOAD, STORE, LOADH, STOREH, LOADB, STOREB, PUSH, POP) uzrokovat će strukturni hazard.

- Te naredbe imati će fazu izvođenja koja se ne može preklapati s fazom dohvata sljedeće naredbe te će njihovo izvođenje efektivno trajati dva perioda. Zato ćemo te naredbe svrstati u grupu **dvociklusnih** naredaba.

Izvođenje memorijskih naredaba

- **Memorijske naredbe LOAD i STORE:**
  - Naredba LOAD prilikom izvođenja mora pročitati podatak iz memorije, a naredba STORE ga mora upisati u memoriju (sve isto vrijedi i za naredbe LOADH,LOADB, STOREH, STOREB)
  - Pristup memoriji pri izvođenju LOAD/STORE **ne može** se odvijati istodobno s dohvatom sljedeće naredbe iz iste memorije
  - Zato se, nakon prepoznavanja naredbe LOAD/STORE, **onemogućuje** rad razine dohvata u sljedećem ciklusu (mjehurić) te će biti aktivna samo razina izvođenja
  - Na kraju izvođenja se zato **omogućuje** rad razine za dohvat u sljedećem ciklusu
  - Međutim, za vrijeme tog (odgođenog) dohvata bit će neaktivna razina za izvođenje (mjehurić prelazi iz razine dohvata u razinu izvođenja), jer nema dohvaćene naredbe koja bi se mogla izvoditi

Dvociklusne naredbe: dohvat STORE

Prva polovica perioda:

- Postavljanje adrese na adresnu sabirnicu PC → AR

Druga polovica perioda *:

• Čitanje naredbe	(AR) → IR
• Dekodiranje naredbe	dekodiranje
• Dohvat adrese** i slanje prema ALU	(ext → ALU) ili (Rx, ext → ALU)
• Prosljeđivanje adrese ili izračun adrese**	(ALU prosljeđuje) ili (ALU zbraja)
• Postavljanje PC za sljedeću naredbu	PC+4 → PC
• onemogući dohvat u sljedećem ciklusu	

---

* Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran signal WAIT

** Ovisno o načinu adresiranja u naredbi

Dvociklusne naredbe: izvođenje STORE

Prva polovica perioda:

- Postavljanje adrese na adr. sabirnicu — ALU_OUT → AR
- Podatak na sab. podataka (uz shuffle) — REG_B → DR

Druga polovica perioda *:

- omogući dohvat u sljedećem ciklusu

*Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran WAIT signal

Na WEBu se nalazi ispravak opisa izvođenja naredbe STORE iz knjige !!!
(u knjizi je pogrešno ALU_OUT → AR bilo napisano u ciklusu dohvata)

Primjer: STOREB R1, (R2+3ABC)

Razina dohvata:
Prva polovina periode CLOCK-a:
PC -> AR
Druga polovina periode CLOCK-a:
(AR) -> IR,
dekodiranje
ext 3ABC i R2 -> ALU
ALU: izvodi zbrajanje
PC +4 -> PC
onemogući dohvat u sljedećem ciklusu

Razina izvođenja:
Prva polovina periode CLOCK-a:
ALU -> AR
R1 -> DR (uz shuffle)

Druga polovina periode CLOCK-a:
omogući dohvat u sljedećem ciklusu

Primjer: LOAD R0, (R1+300)

Razina dohvata:
Prva polovina periode CLOCK-a:
PC -> AR
Druga polovina periode CLOCK-a:
(AR) -> IR,
dekodiranje
ext 300 i R1 -> ALU
ALU: izvodi zbrajanje
PC +4 -> PC
onemogući dohvat u sljedećem ciklusu

Razina izvođenja:
Prva polovina periode CLOCK-a:
ALU -> AR
Druga polovina periode CLOCK-a:
(AR) -> DR
DR -> R0
omogući dohvat u sljedećem ciklusu

Dvociklusne naredbe: dohvat PUSH

(Dohvat je sličan kao i kod LOAD/STORE samo se za adresu uzima R7-4)

Prva polovica perioda:

- Postavljanje adrese na adresnu sabirnicu — PC → AR

Druga polovica perioda *:

- Čitanje naredbe — (AR) → IR
- Dekodiranje naredbe — dekodiranje
- Dohvat adrese i slanje prema ALU — R7, 4 → ALU
- ALU — ALU oduzimanje
- Postavljanje PC za sljedeću naredbu — PC+4 → PC
- onemogući dohvat u sljedećem ciklusu

* Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran signal WAIT

Dvociklusne naredbe: izvođenje PUSH

Prva polovica perioda:

- Postavljanje adrese na adresnu sabirn. ALU_OUT → AR
- Osvježavanje SP ALU_OUT → R7
- Podatak na sabirnicu podataka REG_B → DR

Druga polovica perioda *:

- omogući dohvat u sljedećem ciklusu

* Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran signal WAIT

Upravljački hazard

- Drugi od tri ranije spomenuta hazarda je **upravljački hazard**. Ovaj hazard dešava se kad naredba koja se nalazi u protočnoj strukturi i spremna je za izvođenje nije naredba koja se u stvari treba izvesti
- Ovaj hazard događa se kod izvođenja naredaba grananja kad je procesor već učitao sljedeću naredbu i pripremio se za njeno izvođenje, ali zbog grananja program treba nastaviti s izvođenjem naredbe na nekoj drugoj adresi
- Zbog toga se ovaj hazard naziva još i hazardom grananja
- Naredbe koje uzrokuju ovaj hazard kod FRISC-a su upravljačke naredbe

Uvjetna upravljačka naredba

- uvjetna upravljačka naredba ponaša se kao:
  - dvociklusna kada je uvjet zadovoljen (jer dolazi do pojave hazarda i umeće se mjehurić u protočnu strukturu),
  - jednociklusna kada uvjet nije zadovoljen

Upravljačke naredbe: dohvat JP

Prva polovica perioda:

- Postavljanje adrese na adresnu sabirnicu PC → AR

Druga polovica perioda *:

- Čitanje naredbe (AR) → IR
- Dekodiranje naredbe dekodiranje
- Ispitivanje UVJETA
- Postavljanje PC za sljedeću naredbu PC+4 → PC
- Ako je UVJET istinit onemogući dohvat u sljedećem ciklusu

*Ovi koraci izvode se u slučaju da je memorija dovoljno brza i da nije bio aktiviran WAIT signal

Upravljačke naredbe: izvođenje JP

OVO SE IZVODI SAMO AKO JE UVJET BIO ISTINIT!

Prva polovica perioda:

- Nema aktivnosti

Druga polovica perioda:

- Adresa skoka u PC adr → PC
- omogući dohvat u sljedećem ciklusu

Upravljačke naredbe: JR

- Dohvat i izvođenje isto kao i JP osim što se u periodu izvođenja računa adresa skoka:

  PC+ext → PC

Upravljačke naredbe: CALL, RET, HALT

- Ove naredbe također su jednociklusne ako je zadan uvjet za izvođenje i on nije istinit, a dvociklusne su ako su bezuvjetne ili je zadani uvjet istinit

- DZ: proučite iz knjige korake pri dohvatu i izvođenju naredbe CALL

Podatkovni hazard

- **Podatkovni hazard** javlja se kod izvođenja naredaba u protočnoj strukturi kada se naredba ne može izvesti jer podaci potrebni za njeno izvođenje još nisu spremni

- Kod FRISC-a nema pojave podatkovnog hazarda pa ćemo primjer ovog hazarda proučiti kod procesora ARM

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** 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** 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

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

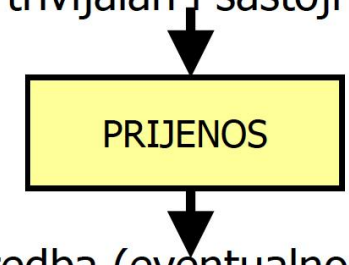
- Glavne karakteristike **programskog prijenosa** su:
  - Prijenos se obavlja **pod upravljanjem programa**, tj. prijenos obavlja procesor (upravljan programom)
  - Svi podatci koji se prenose "**prolaze kroz procesor**"
  - Programski prijenos je **sporiji** od sklopovskog prijenosa
  - Procesor dio 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** 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 UI jedinice
- Dijagram toka je trivijalan i sastoji se samo od prijenosa:



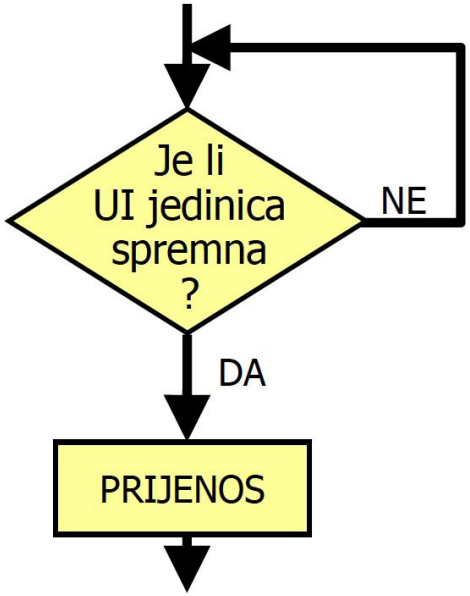
- Prijenos je jedna naredba (eventualno više njih) za čitanje iz VJ ili za pisanje u VJ

Bezuvjetni prijenos

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

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

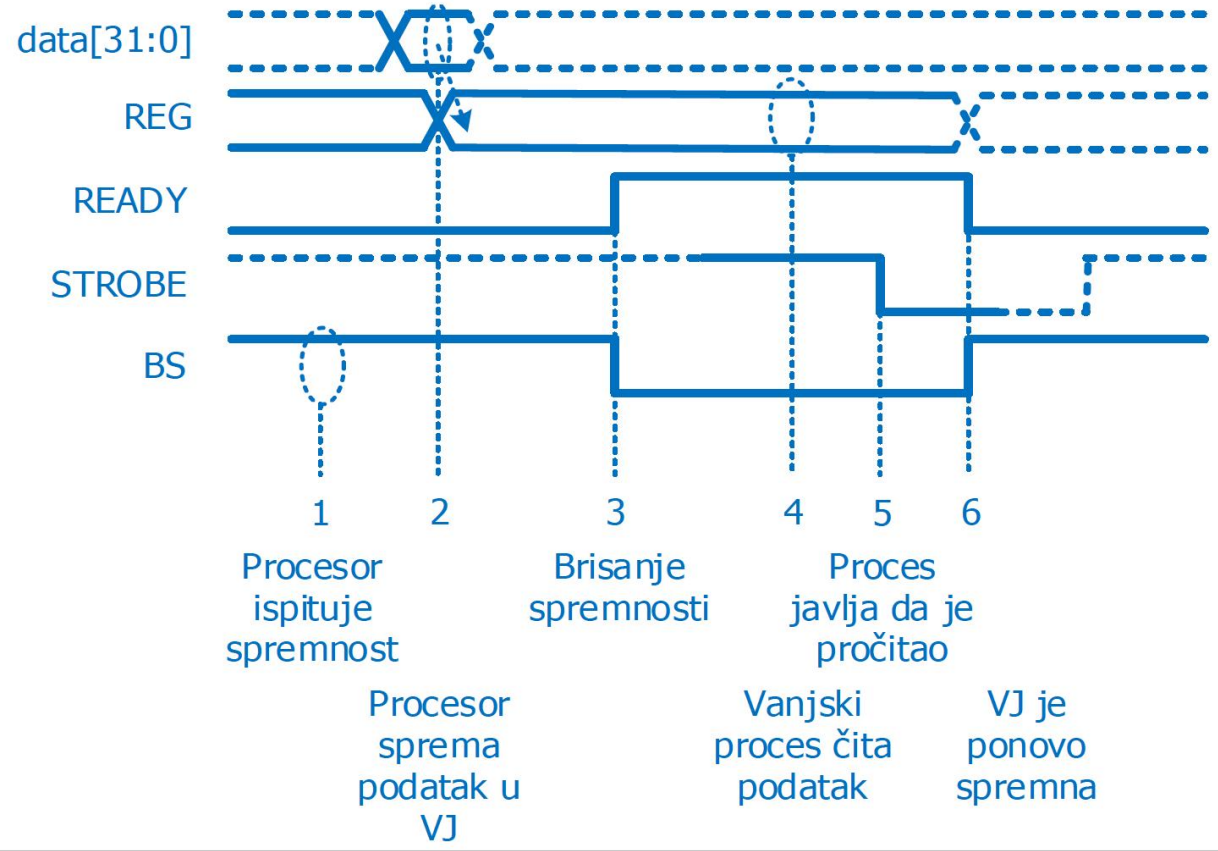


Uvjetni prijenos

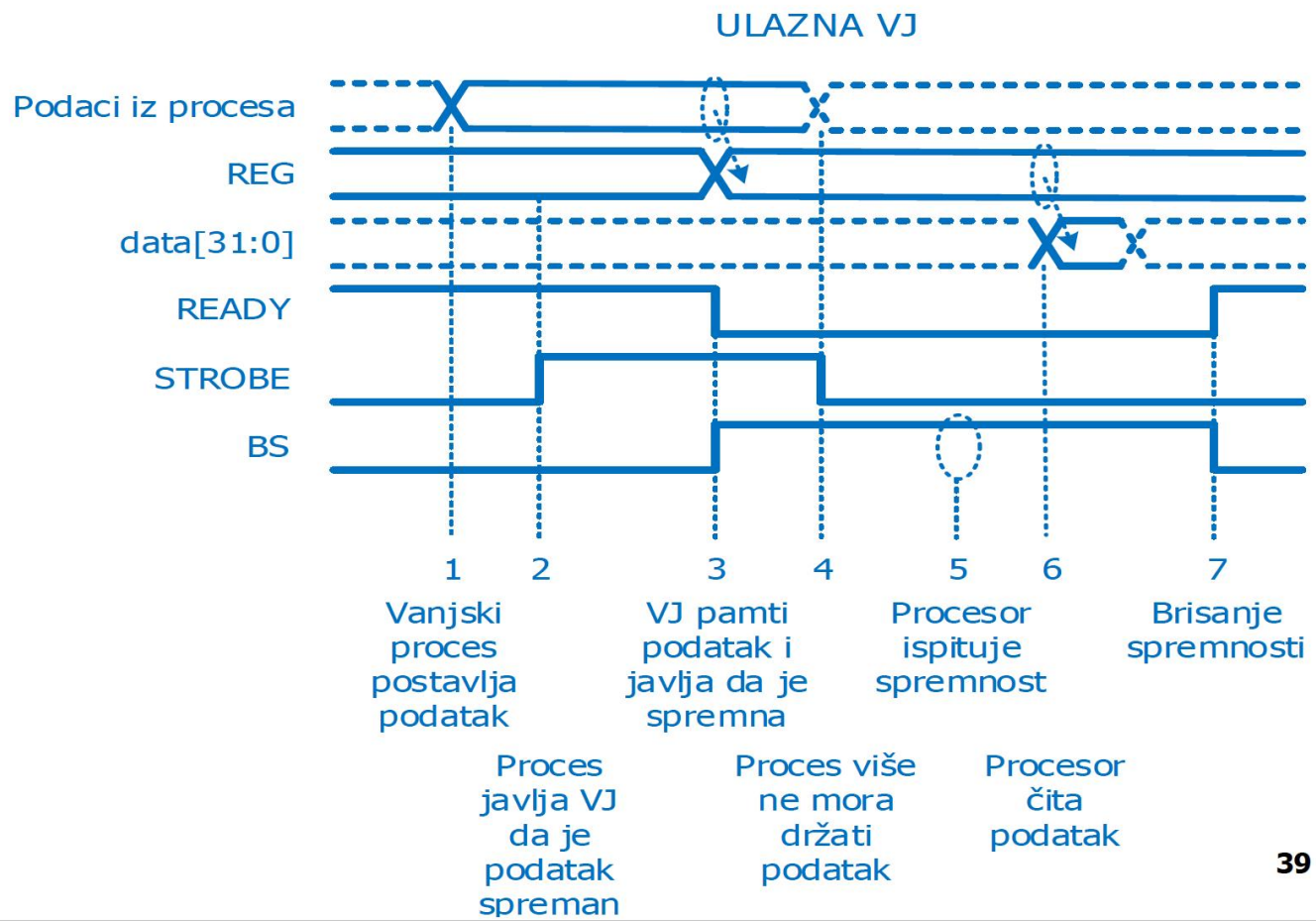
- Glavni **nedostatak** uvjetnog prijenosa:
  - Budući da je procesor tipično puno brži od vanjskih uređaja, može se dogoditi da procesor puno vremena gubi na čekanje da VJ postane spremna

- 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

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

Prekidni sustav FRISC-a

- Prekidni priključci FRISC-a su na sabirnici int[1:0]
  - int[0] - maskirajući prekid (označava se i s INT)
  - int[1] - nemaskirajući prekid (označava se i s NMI)

- **Maskirajući prekid** (maskable interrupt) možemo programski zabraniti ili onemogućiti (maskirati). Ovdje se radi o dozvoljavanju ili zabranjivanju **prihvaćanja prekida** od strane procesora (ne o dozvoljavanju ili zabranjivanju postavljanja zahtjeva od strane VJ)
- **Nemaskirajući prekid** (nonmaskable interrupt) ne možemo ga zabraniti

- Nemaskirajući prekid je **višeg prioriteta** od maskirajućeg
- Maskirajući prekid je **inicijalno zabranjen** (nemaskirajući je, naravno, dozvoljen)

Prekidni sustav FRISC-a

- Prekid se maskira pomoću prekidne zastavice GIE (global interrupt enable) u registru stanja SR:

- **GIE**
  - 0: prihvaćanje maskirajućeg prekida zabranjeno
  - 1: prihvaćanje maskirajućeg prekida dozvoljeno

Prekidni sustav FRISC-a

- FRISC ima zastavicu **IIF** (internal interrupt flag) koja nije u registru SR:
  - ova zastavica nije dostupna programeru, a koristi se kod nemaskirajućeg prekida NMI
  - početno stanje IIF je 1
    - dok se ne obrađuje NMI, IIF je u stanju 1
  - Čim se prihvati NMI, IIF se automatski prebacuje u stanje 0
    - IIF se automatski vraća u stanje 1 po povratku iz NMI
  - dok se obrađuje NMI (na temelju stanja IIF=0):
    - novi zahtjev NMI se ne prihvaća
    - zahtjevi sa INT se ne prihvaćaju

Prekidni sustav FRISC-a

- **Ispitivanje** prekida kod FRISC-a:
  - Postojanje prekida ispituje se na kraju perioda CLOCK-a
    - Naredba koja je u razini izvođenja se izvodi do kraja
  - Ispitivanje i prihvaćanje prekida ovisi o stanju zastavica i trenutačnim zahtjevima za prekid:
    - Ako je IIF=0, prekidi se ne prihvaćaju
    - U suprotnom, ako je NMI prisutan, on se prihvaća, a ako NMI nije prisutan, onda se ispituje maskirajući prekid
    - Ako je GIE=0, maskirajući prekid se ne prihvaća
    - U suprotnom se maskirajući prekid prihvaća

Prekidni sustav FRISC-a

- **Prihvaćanje nemaskirajućeg** prekida kod FRISC-a:
  - Briše se IIF (zabranjivanje svih daljnjih prekida)
  - Sprema se PC na stog
  - Skok u prekidni potprogram na adresi C_{16} (C_{16} → PC)
- Komentari:
  - Dojava VJ da je prihvaćen zahtjev za prekid obavlja se programski
  - Prekidni potprogram mora biti uvijek na memorijskoj adresi 12_{10} (tj. $0C_{16}$)

Prekidni sustav FRISC-a

- **Prihvaćanje maskirajućeg** prekida kod FRISC-a:
  - Briše se GIE (zabranjivanje daljnjih maskirajućih prekida)
  - Sprema se PC na stog
  - Dohvat **adrese** prekidnog potprograma (tzv. prekidnog vektora) s memorijske lokacije na adresi 8 i skok u prekidni potprogram, tj. (8) → PC
- Komentari:
  - dojava o prihvaćanju prekida programski u prekidnom potprogramu
  - Prekidni vektor omogućuje postavljanje prekidnog potprograma na bilo koju adresu u memoriji, ali zahtijeva jedan ciklus čitanja više za dohvat prekidnog vektora. Prekidni vektor mora biti zapisan na adresi 8

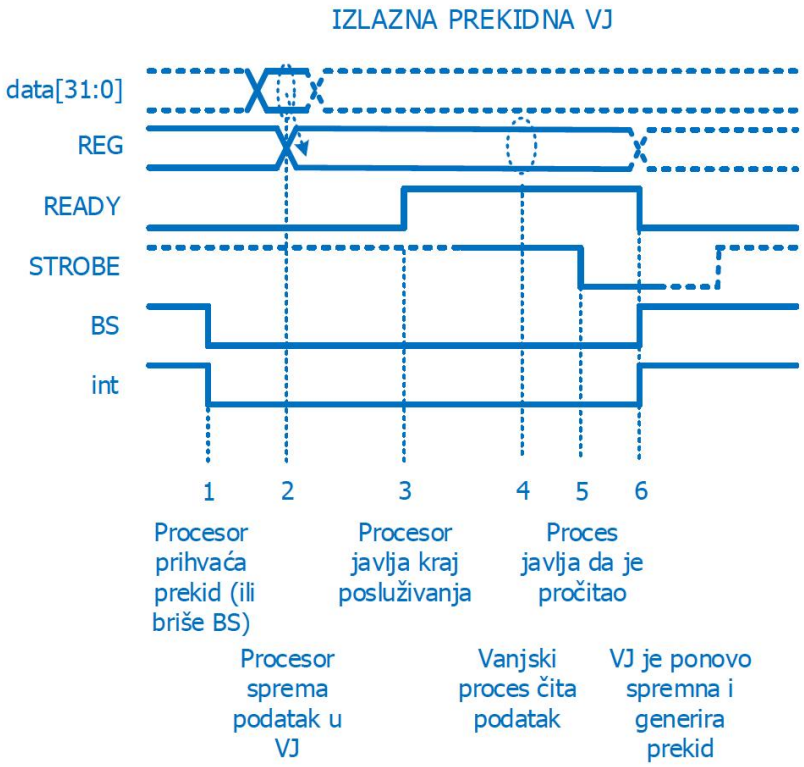
Prekidni sustav FRISC-a

- Naredbe za povratak iz potprograma rade kao i običan RET, ali dodatno dozvoljavaju prekid koji je FRISC bio automatski zabranio kod prihvaćanja prekida (drugim riječima, obnavljaju stanje prekidne zastavice GIE odnosno IIF):
  - Za **maskirajući prekid** je prije prihvaćanja prekida vrijedilo GIE=1, a u trenutku prihvaćanja maskirajućeg prekida se GIE automatski obriše
  - RETI (**RET**urn from maskable **I**nterrupt) obnavlja stanje GIE=1
  - Za **nemaskirajući prekid** je prije prihvaćanja prekida vrijedilo IIF=1, a u trenutku prihvaćanja nemaskirajućeg prekida se IIF automatski obriše
  - RETN (**RET**urn from **N**onmaskable interrupt) obnavlja stanje IIF=1

Osnovna građa prekidne UI jedinice

- Nakon što bistabil stanja postane 1 (kad VJ postane spremna), automatski se postavlja zahtjev za prekid (uz pretpostavku da je dozvoljeno postavljanje prekida - u suprotnom bistabil stanja ne utječe na stanje prekidnog priključka)
- Brisanje bistabila stanja:
  - **uklanja zahtjev za prekid** pa ima ulogu dojave o prihvaćanju zahtjeva za prekid (radi se na početku prekidnog potprograma)
  - **ne omogućava nastavak komunikacije s vanjskim procesom** (za razliku od brisanja bistabila stanja kod uvjetne jedinice)
- Nastavak komunikacije s vanjskim procesom moguć je tek nakon što se VJ dojavi da je njen prekid obrađen
  - to znači da tek nakon toga VJ može ponovno postati spremna i postaviti novi prekid (radi se na kraju prekidnog potprograma)

Vrem. dijagram za izlaznu prekidnu VJ



Vrem. dijagram za ulaznu prekidnu VJ