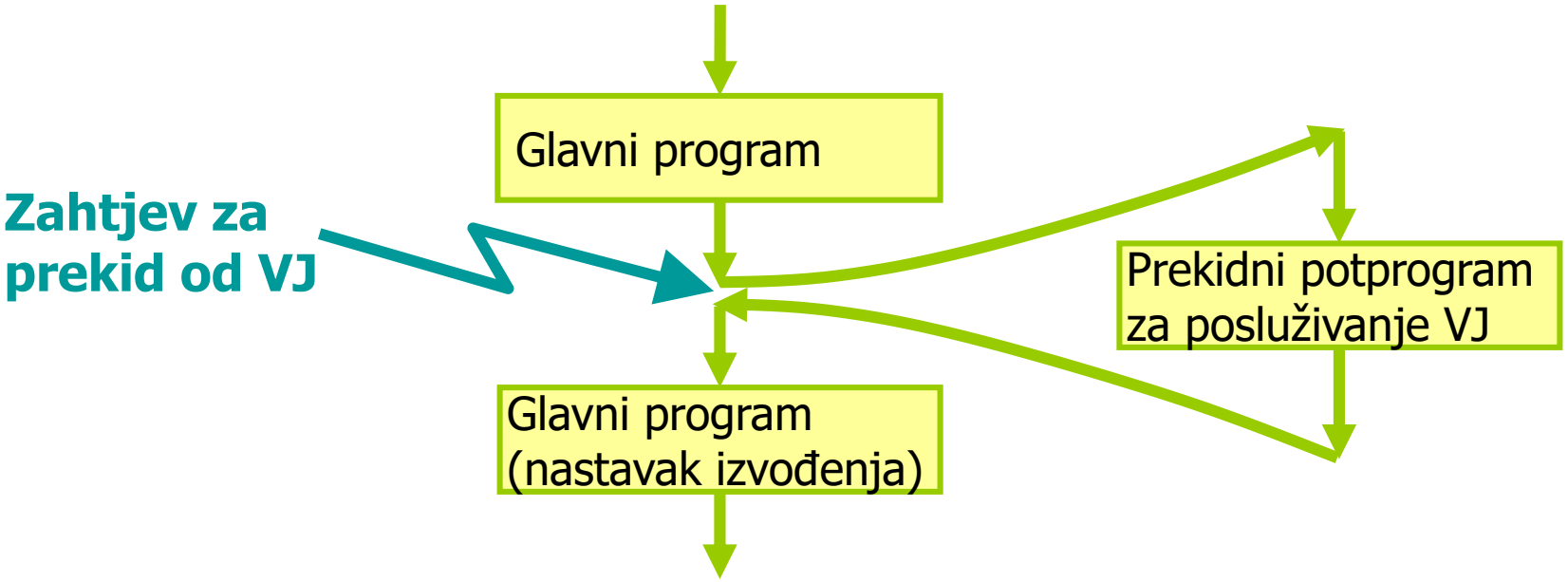
Prekidni prijenos

Prekidni prijenos

- Glavna značajka prekidnog prijenosa je da **UI jedinica samostalno dojavljuje** svoju spremnost procesoru koji za to vrijeme normalno izvodi neki program
  - Spremnost se dojavljuje zahtjevom za prekid (engl. interrupt request)



Iz dijagrama toka vidi se da zahtjev za prekidom (ili kraće prekid) može doći u bilo kojem trenutku izvođenja glavnog programa

Prekidni prijenos

- **Prekidni prijenos** rješava:
  - problem gubitka i uvišestručenja podataka (koji postoji kod bezuvjetnog prijenosa)
  - gubitka vremena na čekanje spremnosti (koji postoji kod uvjetnog prijenosa)
- Prekidni prijenos je učinkovitiji od uvjetnog (u smislu količine dodatnog posla kojeg procesor u jedinici vremena može obaviti uz komunikaciju s VJ), ali ipak nije tako učinkovit kao bezuvjetni (zato što se vrijeme se troši na prihvaćanje zahtjeva za prekid, odlazak u prekidni potprogram i povratak iz prekidnog potprograma)
- Prekidna jedinica građena je slično uvjetnoj, ali je ipak nešto složenija
- Prekidni prijenos koristimo u istim slučajevima kad i uvjetni, ali kad nam je važno da procesor može izvoditi neki program bez usporenja zbog čekanja spremnosti VJ

Prekidni sustavi procesora

- Prekidni sustavi jako se razlikuju od procesora do procesora
  - zato nećemo objašnjavati sve moguće varijante prekidnih sustava
  - orijentirat ćemo se na konkretni prekidni sustav procesora FRISC (i kasnije procesora ARM)

- Prekidni sustav definira sljedeće:
  - koliko prekidnih priključaka procesor ima i koji su im prioriteti
  - kako procesor potvrđuje UI jedinici da je prihvatio zahtjev za prekid
  - kako se određuje adresa prekidnog potprograma
  - može li se prekidni potprogram ponovno prekinuti i kako
  - kako se prepoznaje UI jedinica koja je izazvala prekid
  - kako se procesoru može dozvoliti ili zabraniti prihvaćanje prekida
  - kako jedinica zna da je njeno posluživanje dovršeno
  - kako se obavlja poziv i povratak iz prekidnog potprograma

Prekidni sustavi procesora

- Načelno ponašanje procesora s obzirom na prekide:

1. Procesor izvodi program, a VJ postavlja zahtjev za prekid
2. Procesor izvodi trenutnu naredbu do kraja, tj. ispituje ima li zahtjeva za prekid tek na kraju izvođenja naredbe
3. Ako je u procesoru dozvoljeno prihvaćanje postavljenog prekida, onda procesor prihvaća prekid, a u suprotnom nastavlja s radom
4. Prihvaćanje prekida sastoji se od:
    1. Procesor zabranjuje prihvaćanje daljnjih prekida (osim eventualno prekida jačeg prioriteta ako ih podržava)
    2. Procesor određuje adresu prekidnog potprograma
    3. Procesor pohranjuje registar PC, a često i registar stanja (može pohranjivati i druge registre)
    4. Procesor skače u prekidni potprogram

Prekidni sustavi procesora

- Načelno ponašanje prekidnog potprograma (skraćeno p.p.):

1. Sprema se kontekst (sve registre koje potprogram mijenja, a nisu automatski spremljeni prilikom prihvaćanja prekida)
2. Otkriva se uzročnik prekida (ako ih ima više), tj. otkriva se koja VJ je izazvala prekid *
3. Dojavljuje se VJ da je njen prekid prihvaćen (VJ mora ukloniti zahtjev za prekid) *
4. Poslužuje se VJ
5. Obnavljanje konteksta
6. Ponovno dozvoljavanje prekida **
7. Dojavljuje se VJ da je njezin prekid obrađen (VJ može nastaviti s radom, ponovno postati spremna i zahtijevati prekid) **, ***
8. Izlazak iz prekidnog potprograma i povratak u glavni program na mjesto gdje je bio prekinut

* ovisno o procesoru može se izvesti sklopovski

** ovisno o procesoru može se izvesti sklopovski prilikom koraka 8

*** ovaj korak ne postoji kod svih procesora

Prekidni sustav procesora FRISC

Prekidni sustav FRISC-a

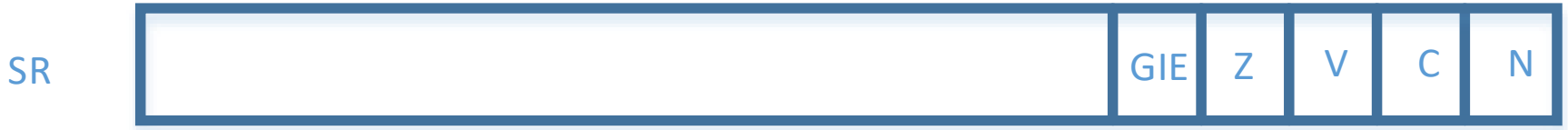
- Prekidni priključci FRISC-a su na sabirnici int[1:0]
  - int[0] - maskirajući prekid (označava se i s INT)
  - int[1] - nemaskirajući prekid (označava se i s NMI)

- **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} ($C_{16} \rightarrow$ 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

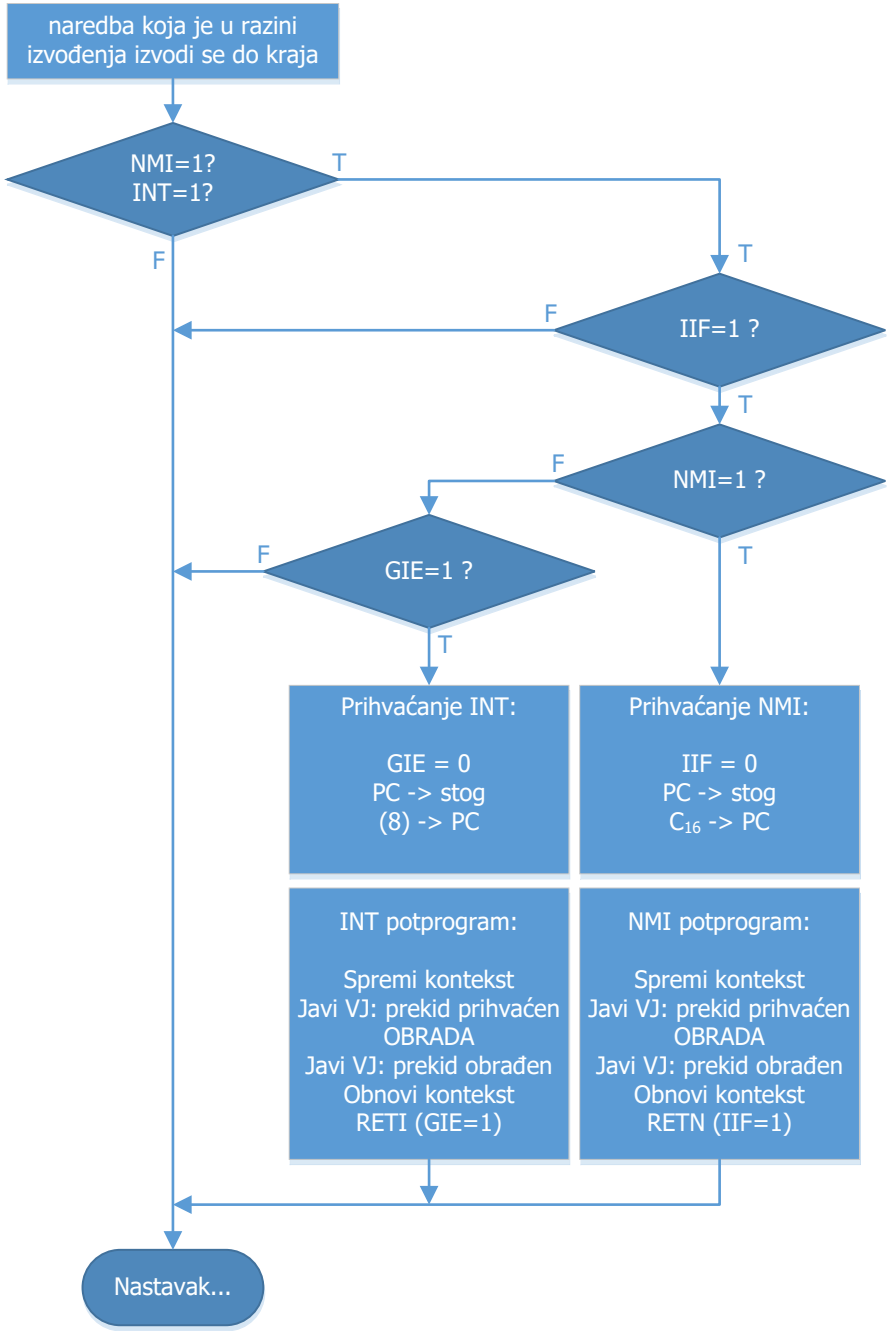
Prekidni potprogram FRISC-a

1. Sprema se kontekst (registre koje potprogram mijenja)
2. Otkriva se uzročnik prekida (ako ih ima više), tj. otkriva se koja VJ je izazvala prekid
3. Dojavljuje se VJ da je njen prekid prihvaćen
4. Poslužuje se VJ
5. Obnavljanje konteksta
6. Dojava VJ da je njezin prekid obrađen
7. Povratak iz potprograma s dozvoljavanjem prekida
    - naredbom RETI za maskirajući
    - naredbom RETN za nemaskirajući

* Koraci 5. i 6. mogu se izvesti i u obratnom redoslijedu

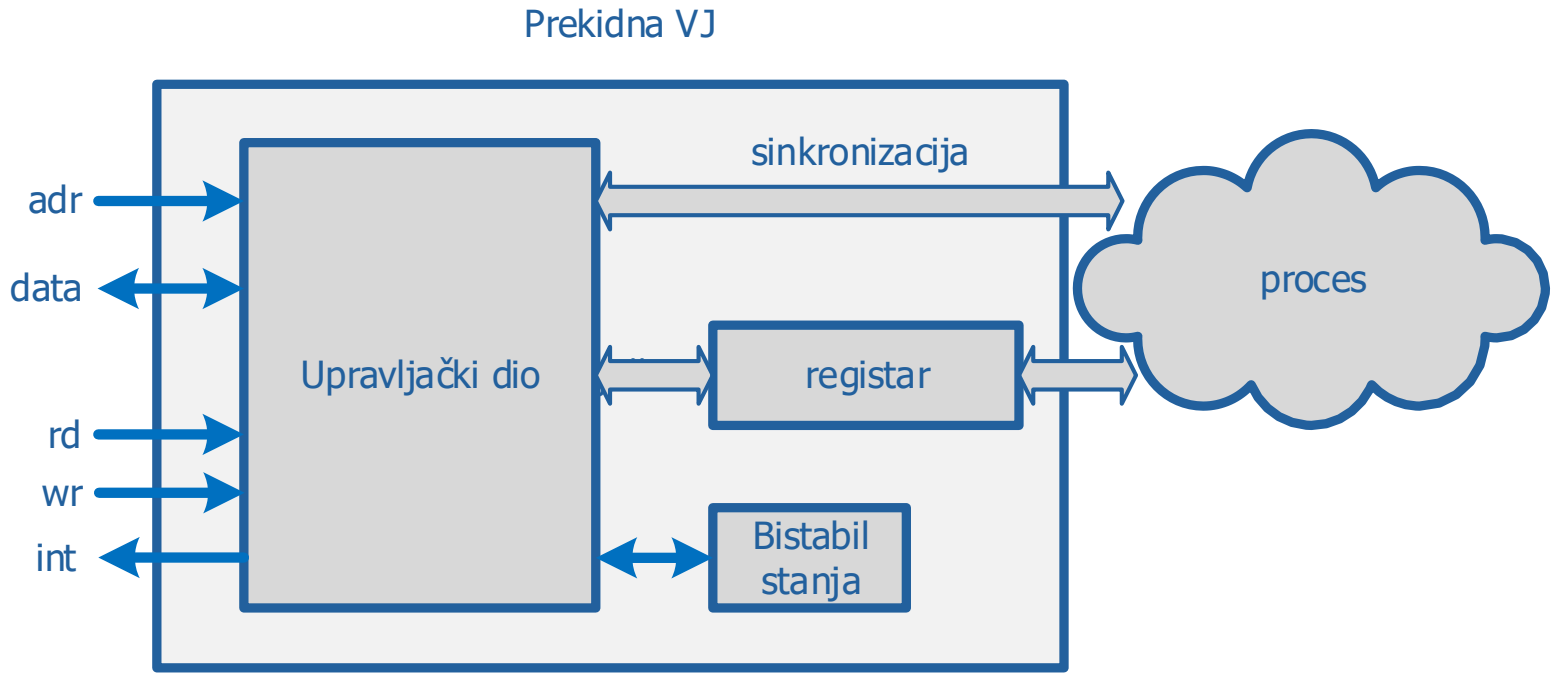
Prekidni sustav FRISC-a

- 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


naredba koja je u razini izvođenja izvodi se do kraja
NMI=1?
INT=1?
T
F
T
F
IIF=1 ?
T
F
NMI=1 ?
T
F
GIE=1 ?
T
Prihvaćanje INT:
GIE = 0
PC -> stog
(8) -> PC
Prihvaćanje NMI:
IIF = 0
PC -> stog
C16 -> PC
INT potprogram:
Spremi kontekst
Javi VJ: prekid prihvaćen
OBRADA
Javi VJ: prekid obrađen
Obnovi kontekst
RETI (GIE=1)
NMI potprogram:
Spremi kontekst
Javi VJ: prekid prihvaćen
OBRADA
Javi VJ: prekid obrađen
Obnovi kontekst
RETN (IIF=1)
Nastavak...

Osnovna građa prekidne UI jedinice

- Najjednostavnija građa opće prekidne UI jedinice može se prikazati sljedećom blok shemom (sve je slično kao kod uvjetne UI jedinice)



Osnovna građa prekidne UI jedinice

- Prekidna VJ može biti spremna ili nespremna kao i uvjetna VJ:
  - Uvjetna VJ je "pasivna": procesor treba ispitivati spremnost što znači da je cijeli tijek prijenosa pod upravljanjem programa
  - Prekidna VJ je "aktivna": kad postane spremna, sama od procesora zahtijeva posluživanje, postavljajući zahtjev za prekid.
- Prekidnoj VJ se programski može zabraniti ili dozvoliti da postavlja prekid kad postane spremna (to je **različito** od dozvoljavanja i zabranjivanja prihvaćanja prekida u procesoru)
  - Obično VJ kojoj se zabrani postavljanje prekida i dalje normalno radi te je se može posluživati kao uvjetnu VJ
- Zabranjivanje postavljanja prekida ima učinak zaustavljanja VJ u prekidnom načinu rada

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)

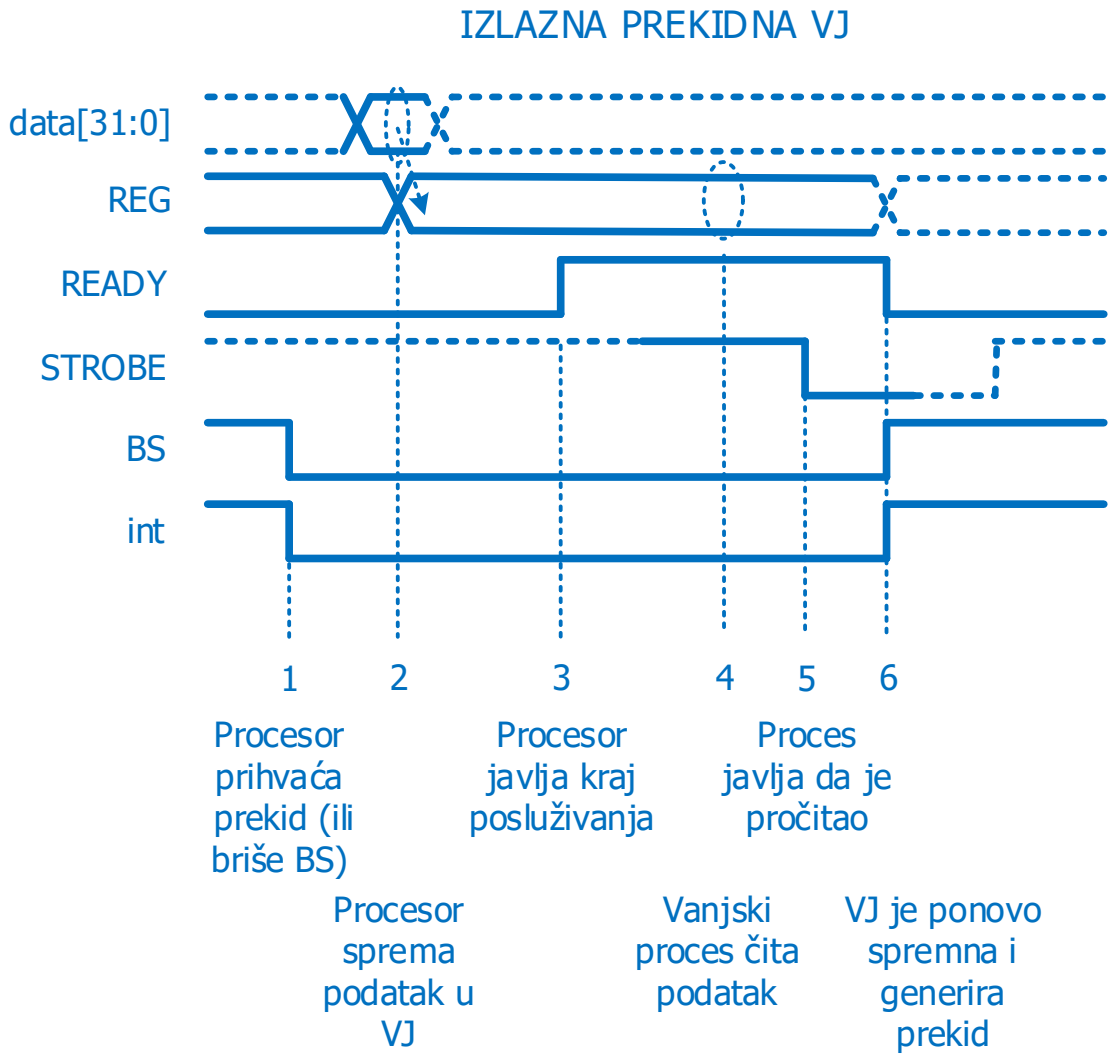
- Brisanje bistabila stanja:
  - **uklanja zahtjev za prekid** pa ima ulogu dojave o prihvaćanju zahtjeva za prekid (radi se na početku prekidnog potprograma)
  - **ne omogućava nastavak komunikacije s vanjskim procesom** (za razliku od brisanja bistabila stanja kod uvjetne jedinice)

- 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)

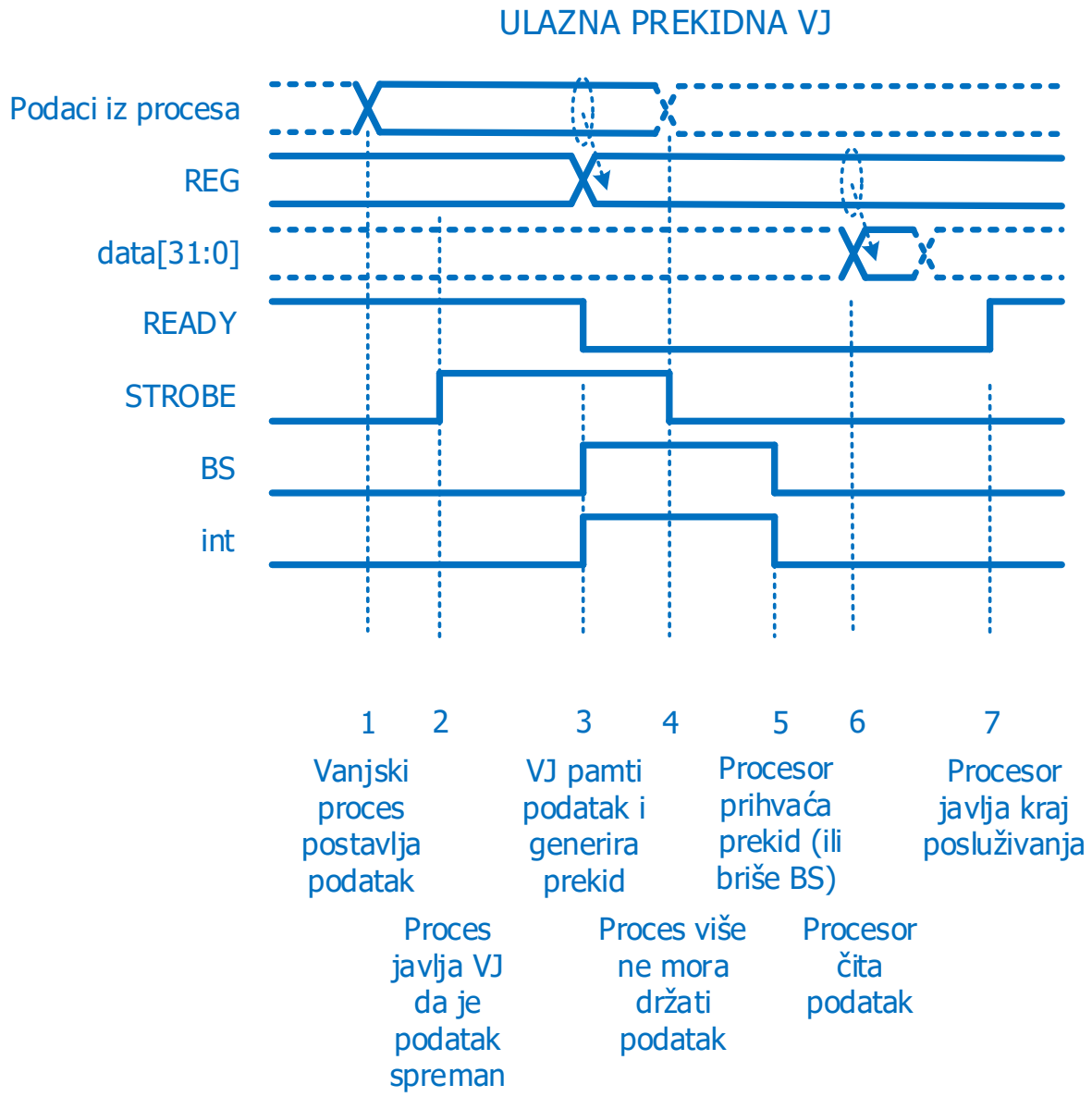
Osnovna građa prekidne UI jedinice

- Prekidna VJ zauzimat će **četiri uzastopne 32-bitne lokacije (* kod konkretnih VJ raspored može biti drugačiji):**
  - Na **prvoj lokaciji** se čita ili piše podatak
  - Na **drugoj lokaciji** se pristupa bistabilu stanja:
    - Čita se trenutačni sadržaj bistabila (ispitivanje stanja)
    - Briše se bistabil operacijom upisa bilo kojeg podatka (poslani podatak se zanemaruje)
  - Na **trećoj lokaciji** se upisom bilo kojeg podatka (poslani podatak se zanemaruje) dojavljuje da je prekid obrađen
  - Pomoću **četvrte lokacije** upravlja se postavljanjem prekida:
    - Upis 0 zabranjuje, a upis 1 dozvoljava postavljanje zahtjeva za prekid
    - Čitanje vraća trenutačnu o(ne)mogućenost postavljanja prekida
    - Inicijalno ćemo pretpostaviti da je dozvoljeno postavljanje zahtjeva za prekid

Vrem. dijagram za izlaznu prekidnu VJ



Vrem. dijagram za ulaznu prekidnu VJ



Prekidni prijenos - Primjeri

FRISC treba primiti 100_{16} podataka od prekidne VJ spojene na NMI. Adresa VJ je FFFF3000. Primljene podatke treba spremati u memorijski blok podataka na adresi 1000, samo ako su pozitivni.

Nakon primitka svih podataka treba zaustaviti rad VJ i rad programa.

```
VJ_DATA EQU  0FFFF3000
VJ_STAT EQU  0FFFF3004
VJ_IEND EQU  0FFFF3008
VJ_STOP EQU  0FFFF300C

        ORG  0
        MOVE  10000, R7   ; početak izvođenja
        JP    GLAVNI      ; skoči na početak glavnog programa

        ORG  0C          ; adresa p.p. za NMI
        PUSH  R0          ; spremi kontekst
        PUSH  R1
        MOVE  SR,R0
        PUSH  R0

        STORE R0,(VJ_STAT) ; briši BS (dojavi prihvaćanje prekida)
        LOAD  R0,(ADR_PODAT)
        LOAD  R1,(VJ_DATA) ; primi podatak i...
        OR    R1,R1,R1
        JR_M  NEMOJ        ;...ako je pozitivan

SPREMI  STORE R1,(R0)       ;...spremi ga u blok...
        ADD   R0,4,R0
        STORE R0,(ADR_PODAT)
```

```
NEMOJ    LOAD  R0,(BROJAC)  ; provjera brojača...
         SUB   R0,1,R0      ;... primljenih podataka
         STORE R0,(BROJAC)
         JR_NZ VAN          ;ima jos podataka->VAN

STOP     STORE R0,(VJ_STOP) ;zaustavi VJ i procesor
         MOVE  1,R0
         STORE R0,(PROC_HALT)

VAN      STORE R0,(VJ_IEND) ; dojavi kraj posluživanja

         POP   R0           ; obnova konteksta
         MOVE  R0,SR
         POP   R1
         POP   R0
         RETN               ; povratak i IIF=1
```

```
GLAVNI
PETLJA    LOAD R0,(PROC_HALT) ; "koristan posao"
          OR   R0,R0,R0
          JR_Z PETLJA         ; nastavi ako je 0

          HALT

PROC_HALT DW   0              ; oznaka za glavni program
                              ; 0 = nastavi rad, 1 = zaustavi procesor
BROJAC      DW   100  ; brojač prenesenih podataka
ADR_PODAT   DW   1000 ; adresa za spremanje u blok
```

Prekidni prijenos - Primjeri

FRISC treba poslati 100_{16} 16-bitnih podataka iz bloka memorije na adresi 1000 na prekidnu VJ na adresi FFFF0000. VJ je spojena na INT. Nakon prijenosa cijelog bloka treba zaustaviti rad prekidne VJ, a glavni program treba nastaviti s radom.

```
SEND  EQU    0FFFF0000
IACK  EQU    0FFFF0004
IEND  EQU    0FFFF0008
STOP  EQU    0FFFF000C

          ORG    0
          MOVE    10000, R7   ; početak izvođenja
          JP      GLAVNI      ; preskakanje vektora

          ; PREKIDNI VEKTOR na adresi 8
          ORG    8
          DW     200    ; adresa p.p.



; GLAVNI PROGRAM
GLAVNI    MOVE   1000, R0       ; adresa podataka
          STORE  R0, (PODATAK)
          MOVE   100, R0        ; brojač podataka
          STORE  R0, (BROJAC)
          ; DOZVOLI PREKID NA INT0
          MOVE   %B 10000, SR

PETLJA    JP    PETLJA       ; "koristan posao"
```

```
; PREKIDNI POTPROGRAM NA ADRESI 200
          ORG   200
          PUSH   R0         ; spremanje konteksta
          PUSH   R1
          PUSH   R2
          MOVE   SR,R0
          PUSH   R0

          STORE  R0, (IACK)    ; prihvaćen prekid

          LOAD   R0, (BROJAC)  ; dohvat varijabli
          LOAD   R1, (PODATAK)

          LOADH  R2, (R1)     ; čitanje iz memorije
          STORE  R2, (SEND)   ; i slanje na VJ

          ADD    R1, 2, R1    ; pomicanje pokazivača
          STORE  R1, (PODATAK)
SUB       R0, 1, R0    ; smanjenje brojača
          STORE  R0, (BROJAC)
          JR_NZ  IMA_JOS
```

```
ZADNJI   MOVE   0, R0        ; ako je zadnji podatak
         STORE  R0, (STOP)   ; zaustavi VJ

IMA_JOS  POP    R0          ; obnavljanje konteksta
         MOVE   R0, SR
         POP    R2
         POP    R1
         POP    R0

         STORE R0, (IEND)   ; kraj posluživanja
         RETI
BROJAC   DW    0           ; varijable za p.p.
PODATAK  DW    0

        ; Podaci iz memorije koji se šalju na VJ
         ORG   1000
         DH    12, 4, 456A, 1, 0AB, 2, 885, ...
```

Prekidni prijenos - Primjeri

- Komentari:
- **U kontekst prekidnog potprograma ulazi i SR*** (osim u rijetkim slučajevima kad se SR ne mijenja u prekidnom potprogramu)

- **Za obične potprograme SR ne ulazi u kontekst**, jer pozivatelj može pretpostaviti da će potprogram promijeniti SR i zato pozivatelj nikada nema u SR-u neko stanje koje će mu trebati nakon povratka iz potprograma (Ako pozivatelju običnog potprograma treba stanje iz SR-a, onda ga pozivatelj treba spremiti).

* Neki procesori automatski spremaju statusni registar prilikom prihvaćanja prekida

Posluživanje više prekidnih VJ

- Do sada smo vidjeli samo najjednostavniji slučaj kad je na procesor spojena jedna prekidna VJ

- Kad postoji više prekidnih VJ, one se mogu posluživati sa ili bez gniježđenja (engl. nesting):
  - **bez gniježđenja** prekidnih potprograma:
    - **dok se poslužuje jedna VJ, drugi prekidi se ne prihvaćaju**
    - jednostavniji slučaj
  - **sa gniježđenjem** prekidnih potprograma:
    - **dok se poslužuje jedna VJ, može se prihvatiti drugi prekid (većeg prioriteta)**
    - kompliciraniji slučaj

Posluživanje više prekidnih VJ

- Svim vanjskim jedinicama treba **dodijeliti različite prioritete**, što se može napraviti:
  - programski (**FRISC za maskirajuće**)
  - sklopovski
    - sam procesor ima više prekidnih priključaka s različitim prioritetima (**FRISC: NMI je prioritetniji od INT**)
    - prioritetni lanac vanjskih jedinica (daisy-chain)
    - jedinica za kontrolu prioriteta (programmable interrupt controller ili priority interrupt controller)

- **Prioriteti imaju dvojaku ulogu**:
  - kod istovremenih prekida određuje se kojoj VJ se prihvaća prekid (služi i za gniježđenje prekida i kad nema gniježđenja)
  - za vrijeme obrade jednog prekida određuje hoće li se prihvatiti novi prekid (služi samo za gniježđenje prekida)

Posluživanje više prekidnih VJ

- Bez obzira kako se poslužuju, **uvijek treba odrediti uzročnike prekida** o čemu ovisi koju VJ ćemo poslužiti

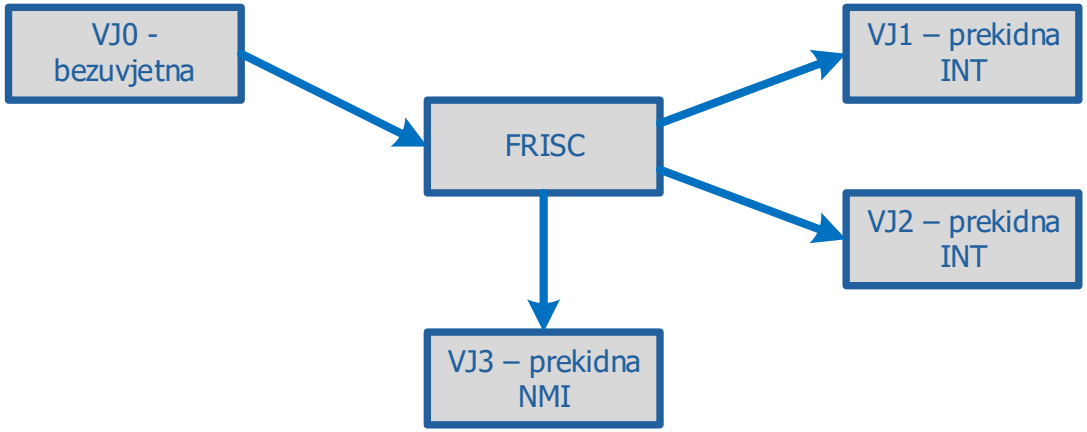
- Ovisno o prekidnom sustavu procesora, moguća su različita rješenja:
  - VJ sklopovski utječe na odabir adrese prekidnog potprograma čime se automatski određuje uzročnik prekida
  - Adresa prekidnog potprograma bira se na temelju ulaznog prekidnog priključka čiji prekid je prihvaćen (**FRISC: NMI ima različitu adresu p.p. od INT**)
  - Programski se određuje koja jedinica je izazvala prekid (**FRISC: ispitivanjem BS**)

Posluživanje više prekidnih VJ

- Ispitivanje uzročnika prekida kod prekida:
  - **ispituje se spremnost VJ** (bistabil stanja)
  - ovo **ne treba miješati s ispitivanjem uvjetnih VJ**, jer se ovdje samo jednom ispita spremnost, tj. nema čekanja da VJ postane spremna niti se obavlja prozivanje
  - **redoslijed ispitivanja definira prioritete VJ**:
    - jedinice koje se prije ispituju imaju veći prioritet

Prekidni prijenos - Primjer

Na FRISC su spojene vj0, vj1, vj2 i vj3 na adresama FFFF0000, FFFF1000, FFFF2000 i FFFF3000. Vj0 je ulazna bezuvjetna, vj1 i vj2 izlazne prekidne jedinice spojene na INT (ne mogu se međusobno prekidati), a vj3 je izlazna prekidna jedinica spojena na NMI. Procesor šalje podatke s vj0 na vj1 i vj2 i broji koliko je podataka poslao. Kad vj3 zatraži prekid, treba joj poslati broj do tada prenesenih podataka.



```
PRIMI0    EQU   0FFFF0000

SALJI1    EQU   0FFFF1000
BS1       EQU   0FFFF1004
POSLUZEN1 EQU   0FFFF1008

SALJI2    EQU   0FFFF2000
BS2       EQU   0FFFF2004
POSLUZEN2 EQU   0FFFF2008

SALJI3    EQU   0FFFF3000
BS3       EQU   0FFFF3004
POSLUZEN3 EQU   0FFFF3008


          ORG    0
          MOVE    10000, SP
          JP      GLAVNI
          ; prekidni vektor za maskirajući prek.
          ORG    8
          DW     100
```

```
        ; prekidni potprogram za nemaskirajući prekid na adresi 0C
        ORG    0C

        PUSH    R0
        STORE R0, (BS3) ; prihvaćen prekid
        LOAD    R0, (BROJAC) ; broj poslanih
        STORE   R0, (SALJI3) ; pošalji na vj3
        POP     R0
        STORE   R0, (POSLUZEN3) ; dojava kraja

        RETN
; Glavni program

GLAVNI   ; dozvoli prekid na INT0
         MOVE  %B 10000, SR
         ; "koristan posao"
PETLJA   JR    PETLJA

        ; brojač poslanih podataka
BROJAC   DW     0
```

```
        ORG  100

        PUSH  R0          ; spremanje
        MOVE  SR, R0      ; konteksta
        PUSH  R0

ISPITAJ LOAD  R0, (BS1)   ; otkrivanje
        AND   R0, 1, R0   ; uzročnika
        JR_NZ P_VJ1       ; prekida
        JR    P_VJ2

VAN     POP   R0
        MOVE  R0, SR      ; obnova
        POP   R0          ; konteksta

        RETI
```

```
P_VJ1 ; dio za posluživanje vj1
        STORE R0, (BS1) ; prihvaćen prekid

        LOAD  R0, (PRIMI0) ; čitaj bezuvjetnu vj0
        STORE R0, (SALJI1) ; šalji na vj1

        LOAD  R0, (BROJAC) ; povećaj
        ADD   R0, 1, R0    ; brojač poslanih
        STORE R0, (BROJAC) ; podataka

        STORE R0, (POSLUZEN1)  ; kraj posluživanja
        JR    VAN              ; povratak

P_VJ2 ; dio za posluživanje vj2
      ; analogno kao i P_VJ1
      ...
```

(PONOVLJENI SLAJD SA NMI PP)

```
          ; prekidni potprogram za nemaskirajući prekid na adresi 0C
          ORG    0C

          PUSH    R0
          STORE R0, (BS3) ; prihvaćen prekid
          LOAD    R0, (BROJAC) ; broj poslanih
          STORE   R0, (SALJI3) ; pošalji na vj3
          POP     R0
          STORE   R0, (POSLUZEN3) ; dojava kraja

          RETN
; Glavni program

GLAVNI    ; dozvoli prekid na INT0
          MOVE  %B 10000, SR
          ; "koristan posao"
PETLJA    JR    PETLJA

          ; brojač poslanih podataka
BROJAC    DW     0
```

Prekidni prijenos - Primjer

- Komentar:
- Vj1 i vj2 se ne mogu međusobno prekidati, ali u prekidnom potprogramu se prvo ispituje vj1 pa će ona biti prioritetnija od vj2 u smislu da će kod istovremenog prekida prva biti poslužena vj1.
- Kod istovremenog prekida dešava se sljedeće:
- Prihvaća se prekid čime se automatski zabrani prihvaćanje daljnjih prekida. U p.p.-u se ustanovi da je vj1 izazvala prekid te se obrađuje njen prekid. Cijelo to vrijeme vj2 zahtjeva prekid, ali je prihvaćanje prekida u procesoru zabranjeno i prekid od vj2 se ne prihvaća: kažemo da je prekid "na čekanju" (tzv. pending interrupt).
- Nakon povratka iz prekidnog potprograma od vj1, doći će do omogućavanja prekida (naredba RETI). Tada će prekid od vj2 konačno biti prihvaćen te će se skočiti u prekidni potprogram koji će tada poslužiti vj2.

Prekidni prijenos - Primjeri

- DZ: Proučiti dodatne primjere iz knjige i zbirke